VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra Informatiky

IS pořadů Hvězdárny a planetária Johanna Palisy

-

Calendar Programme IS for the Johann Palisa Observatory and Planetarium

2009 Bc. Vojtěch Kačmařík

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě 7. května 2009 . . . . . .

Touto cestou bych rád poděkoval vedoucí diplomové práce, paní Doc. RNDr. Janě Šarmanové, CSc., především za konzultace a odborné rady, které jsem pak mohl zúročit při psaní tohoto textu.

Abstrakt

Moderní informační technologie se v dnešní době rozvíjejí velmi rychlým tempem a důsledkem tohoto jevu je také přirozená reakce okolí na tuto skutečnost. Svět se díky moderní technice za několik málo let výrazně změnil, a pokud někdo nezareaguje na potřebu mít vlastní IT řešení pro svůj byznys nebo organizaci, tak to pro něj může mít v konečném důsledku hluboké následky. Jedna taková potřeba byla zároveň motivací pro tvorbu následující diplomové práce, která se zabývá analýzou a řešením informačního systému pro Hvězdárnu a planetárium Johanna Palisy VŠB-TU Ostrava. Před samotným začátkem práce existoval požadavek na nový informační systém, který by mohl být využíván pro každodenní práci lidí na hvězdárně a zároveň by jim usnadňoval život. Z této potřeby bylo nutné nějakým způsobem definovat samotné zadání pro tvorbu systému, tedy to, proč je systém vůbec vyžadován a co od něj zákazník očekává. Další částí práce bylo zpracovat detailní specifikaci požadavků, která vychází z reálných potřeb zákazníků, v našem případě to je zaměstnanců Hvězdárny a planetária Johanna Palisy. Specifikace byla vytvářena na základě rozhovorů s budoucími uživateli systému, aby se výsledek co nejvíce blížil reálným potřebám. Na základě této specifikace byla poté zpracována analýza celého systému a systém následně implementován. Výsledkem práce by měl být informační systém, který nahradí dosavadní způsob práce ve zmiňované organizaci a zaměstnancům ulehčí jejich práci.

Klíčová slova: Internetové technologie, Informační systém, Microsoft .NET Framework, ASP.NET, AJAX, AJAX.NET, IIS (Internetová informační služba), MS SQL Server

Abstract

The modern information technologies are developed too fast rate in today's period and as a consequence of this situation is nature reaction of the whole society on this fact. The world was changed using these modern technologies in some few years. If someone won't react on the requirement keep own IT solution for some business or organization, so it may has deep consequences to him. One similar requirement was motivation for creation following thesis which puts mind to analysis and solution of the information system for Johann Palisa observatory. Before start working the requirement for new information system existed. The information system should be used for employee's daily work on the observatory and also it should be used to simplify their working life. From this demand it was necessary to refine problem definition for creation new information system, in other words, why is the system being asked and what benefits are expected. Next part of this thesis was creation detailed requirements specification which comes from real needs of customers. The specification was created on the basis of interviews with prospective users of this system to ensure the best quality of resulting system in agreement with real customer needs. According to this specification was sequentially created analysis of the whole system and then the system was implemented. As a result should be a particular information system which will substitute present style of work in mentioned organization and will simplify daily work for all employees in this organization.

Keywords: Internet technologies, Information system, Microsoft .NET Framework, ASP.NET, AJAX, AJAX.NET, IIS (Internet Information Services), MS SQL Server

Seznam použitých zkratek a symbolů

AJAX – Asynchronous Javascript and XML

API – Application Programming Interface

CIL – Common Intermediate Language

CLR – Common Language Runtime

CPU – Central Processing Unit

CSS – Cascading Style Sheets

DFD – Data Flow Diagram

ERD – Entity Relationship Diagram

HaPJP – Hvězdárna a planetárium Johanna Palisy

HTML – HyperText Markup Language

HTTP – Hypertext Transfer Protocol

HTTPS – Hypertext Transfer Protocol Secure

IIS – Internet Information Services

IL – Intermediate Language

IO – Integritní omezení

IS – Information System

JIT – Just-in-Time

MVC – Model-View-Controller

SQL – Structured Query Language

SŘBD – Systém řízení báze dat

SSL – Secure Socket Layer

STD – State Transition Diagram

SW – Software

TCP/IP – Transmission Control Protocol/Internet Protocol

URL – Uniform Resource Locator

WWW – World Wide Web

XHTML – eXtensible HyperText Markup Language

XML – eXtensible Markup Language

Obsah

Obsah přiloženého CD

Abstrakt.pdf
Diplomova_prace.pdf
Graficky_manual\Graficky_manual.pdf
Graficky_manual\Kody_barevnych_papiru.pdf
Home_directory\HaPJP\
Programatorska_dokumentace\Programatorska_dokumentace.pdf
Source_code\Projects\HaPJP\
Source_code\WebSites\HaPJP\
Uzivatelska_dokumentace\IS_HaPJP_part1.pdf
Uzivatelska_dokumentace\IS_HaPJP_part2.pdf

Seznam tabulek

Seznam obrázků

1. Úvod

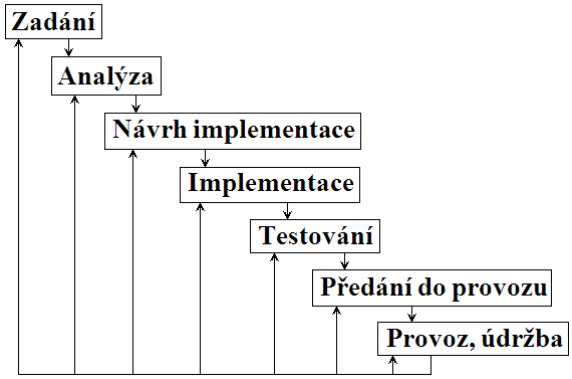
Cílem této diplomové práce je specifikovat a vyvinout nový informační systém, který je požadován od zákazníka, tj. Hvězdárna a planetárium Johanna Palisy VŠB-TU Ostrava. Na základě počátečních požadavků je třeba vytvořit podrobnou analýzu a zpracovat návrh řešení. Podle těchto dílčích částí bude následně systém implementován a později nasazen pro běžné užití.

Na začátku práce je třeba shromáždit všechny hlavní požadavky od zákazníka, tyto požadavky přesně definovat a analyzovat. Především jde o zachycení potřeb všech zainteresovaných osob a cílových uživatelů a zjištění, proč tyto definované potřeby existují. Je třeba rozlišit funkční a nefunkční požadavky na nový systém a podle toho dělat další návrh řešení. Pro detailní zjištění požadavků může být využito různých technik, např. rozhovory se zákazníkem, pozorování práce přímo v organizaci, dotazník, tzv. brainstorming aj.

Detailní zpracování analýzy a návrh řešení nového IS bude obsažen v dalších kapitolách této práce, které se budou konkrétně zabývat datovou analýzou, funkční analýzou, návrhem ovládání a komunikace s uživatelem apod. Obsahem těchto kapitol bude například podrobné schéma databáze, popis jednotlivých funkcí a modulů systému a také například grafický design.

Další kapitola se bude zabývat návrhem samotné implementace nového systému. Zde budou řešeny technické problémy, funkce a řešení jednotlivých modulů, technologie, které budou použity, a další problémy související s návrhem. Z těchto informací bude potom vycházet samotné řešení systému, které bude částečně popsáno v další samostatné kapitole. Poslední kapitoly budou obsahovat popis testování systému, informace důležité k nasazení systému v provozním prostředí a tvorbu uživatelské a programátorské dokumentace.

Celá diplomová práce bude korespondovat s životním cyklem informačního systému (viz. Obrázek 1.1) od zadání, specifikace problému, detailní analýzy až k implementaci, testování a nasazení produktu. Výsledkem by měl být nový informační systém, který bude odpovídat požadavkům z Hvězdárny a planetária Johanna Palisy a který by měl být potom používán ke každodenní práci zaměstnanců ve zmiňované organizaci. Systém by měl být uživatelsky přívětivý a měl by zaměstnancům ulehčovat jejich každodení práci. Předpokládaným výsledkem je zde nový informační systém a také spokojený zákazník.



Obrázek 1.1: Životní cyklus vývoje produktu (vodopádový model)

2. Specifikace požadavků

Specifikace požadavků má za úkol získání a přehledné zpracování všech požadovaných vlastností, detailů a všech možných omezení na vytvářený informační systém Hvězdárny a planetária Johanna Palisy (dále IS HaPJP). Po dokončení metod specifikace požadavků budou přehledně zpracovány všechny funkční a nefunkční požadavky na navrhovaný systém. Další kapitola, která bude po specifikaci požadavků následovat, bude detailní analýza a návrh implementace.

Všechny základní body podle specifikace požadavků (viz. [Šarm 1]) byly náplní první schůzky se zadavatelem, která měla za účel základní seznámení s požadavky na nově budovaný IS. Cílem bylo získat co nejvíce základních informací, pochopit funkci systému a důvody jeho zavedení. Pro získání těchto požadavků a odpovědí na předcházející body jsem zvolil formu rozhovoru, kdy zadavatel nejdříve demonstroval svou vizi produktu a následně jsme se snažili společně odpovědět na určité otázky.

2.1 Funkční požadavky

Seznam funkčních požadavků, které jsou výsledkem první schůzky se zadavatelem, je následující:

Nový informační systém by měl nahradit stávající systém evidence objednávek a dalších údajů na Hvězdárně a planetáriu Johanna Palisy (dále jen HaPJP). Systém by měl dále umožňovat plánovat pořady. Všechny tyto informace se dosud uchovávají na tištěných formulářích a neexistuje zde pohodlná editace údajů a vyhledávání v již existujících záznamech. IS by měl tedy poskytovat přehledný výpis objednávek a pořadů v této organizaci. Nový systém by měl být uživatelsky přívětivý, aby se s ním dobře pracovalo, a měl by především nabídnout něco nového oproti stávajícímu systému a také by měl zaměstnancům ulehčovat jejich práci.

IS by měl sloužit uživatelům (zaměstnancům, zákazníkům), kterým bude umožňovat provádět elektronické objednávky a využívat další funkce systému. Základní použití systému je vytvoření pořadu, záznamu o zákazníkovi a následné vytvoření objednávky. Nový systém bude navíc poskytovat vyhledávání v záznamech a tvorbu výstupních sestav a statistických výstupů.

Uživateli IS budou v první fázi zaměstnanci HaPJP. Další fáze bude obsahovat modul, se kterým budou pracovat zákazníci. Zákazníci by měli mít v podstatě možnost provést pouze on-line objednávku. Zaměstnanci HaPJP by mohli využívat všechny funkce systému s výjimkou funkcí určených pouze pro administrátora. Role systému by se daly rozdělit následujícím způsobem:

- Zaměstnanec
  - Administrátor systému
  - Zaměstnanec
- Zákazník
  - Organizace
  - Soukromá osoba

2.1.1 Vstupní data

Požadovanými vstupními daty budou všechny informace důležité k realizaci základních funkcí systému. Tato data připraví zadavatel pro naplnění potřebných databázových tabulek. Seznam evidovaných entit a jejich atributů je následující:

- Aktualita (datum přidání, autor, obsah)
- Faktura (celkový počet dětí, celkový počet dospělých, počet dětí [Hvězdárna | Minigalerie MIRA | Astronomická jednohubka | Pořad | Přednáškový sál – film], počet dospělých [Hvězdárna | Minigalerie MIRA | Astronomická jednohubka | Pořad | Přednáškový sál – film])
- Kategorie (název kategorie, popis)
- Komentář (autor, datum přidání, obsah)
- Objednávka (zákazník, organizace, faktura, termín, objednaný pořad, kategorie, lektor, počet míst, stav, celková délka, poznámka, název filmu, název jednohubky, objednáno [Hvězdárna | Minigalerie MIRA | Astronomická jednohubka | Přednáškový sál – film])
- Omezení termínu (od kdy, do kdy, důvod omezení)
- Organizace (název organizace, typ, adresa)
- Osoba (uživatelská role, jméno, příjmení, adresa, telefonní číslo, emailová adresa)
- Pořad (název pořadu, kategorie, délka, popis pořadu, systémová zkratka)
- Poznámka (osoba, obsah poznámky)
- Program pro veřejnost (termín, pořad, lektor, popis)
- Rezervace (program pro veřejnost, osoba, organizace, počet míst, datum rezervace)
- Služba (zaměstnanec, datum)
- Termín (datum začátku, datum konce, název dne v týdnu, uzávěrka, tisk)
- Zaměstnanec (uživatelská role, jméno, příjmení, adresa, telefonní číslo, emailová adresa, systémová zkratka)

Data, která bude systém evidovat v rámci objednávky, jsou informace o objednaném pořadu, který zákazník objednávku udělal, na jaký termín a také počet objednaných míst. Je možné si navíc objednat astronomickou jednohubku v planetáriu, krátký film promítaný v přednáškovém sále nebo program přímo na hvězdárně. Návštěvníci mohou navštívit i minigalerii MIRA. Objednávka bude zároveň obsahovat její stav, tedy jestli je nová, upravená, potvrzená nebo stornovaná.

Týdenní rozpis bude obsahovat tzv. programy pro veřejnost, u kterých se bude evidovat přednášený pořad a lektor tohoto pořadu. Na tento program pro veřejnost bude potom možné vytvořit rezervaci, která bude evidovat zákazníka, organizaci, počet míst a datum rezervace. Rozdělení pořadů do kategorií je pouze orientační, kategorie se mohou lišit a také objednávka může být pro různé kategorie posluchačů.

Přihlašovací jméno evidované pro zaměstnance HaPJP bude stejné jako do sítě VŠB-TU Ostrava. Není potřeba vytvářet nové přihlašovací jméno. Zaměstnanec bude mít navíc možnost evidovat v systému u zákazníka textové poznámky.

Systém bude evidovat také údaje o termínech, resp. datech, která nemohou být použita pro objednávání, a programy pro veřejnost. Např. prázdniny, státní svátky, dovolené apod. Dále bude systém evidovat tzv. aktuality, resp. novinky, které budou zobrazeny na titulní stránce pouze pro zaměstnance HaPJP. Zaměstnanci budou schopni tyto aktuality do systému přidávat.

2.1.2 Výstupy systému

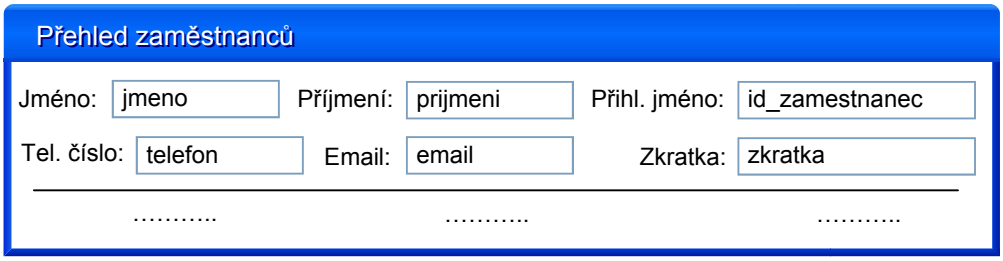
Výstupy, které bude nový IS HaPJP poskytovat, by měly být následující:

- Přehled pořadů (název, zkratka, kategorie)
- Přehled zaměstnanců (jméno, příjmení, telefonní číslo, email, přihlašovací jméno)
- Přehled zákazníků
- Přehled organizací (název, typ, adresa)
- Týdenní přehled objednávek
- Týdenní přehled programů pro veřejnost
- Přehled aktualit (datum, jméno autora, obsah)
- Přehled zakázaných termínů (datum, důvod omezení)
- Přehled kategorií (název kategorie, popis)
- Přehled služeb zaměstnanců (datum, jméno zaměstnance)
- Výstupní tiskové sestavy
- Výstupy z modulu statistické analýzy

Pro výstupní tiskové sestavy byly použity formuláře, které jsou použity na HaPJP doposud, tj. týdenní přehled objednávek a programů pro veřejnost, evidence počtu zúčastněných osob na konkrétní objednávky a objednávkový formulář se všemi důležitými údaji o objednávkách a zákaznících. Sestavy jsou tvořeny na základě těchto vzorových formulářů (viz. Přílohy A).

Další výstupy ze systému budou obsahovat data týkající se jednotlivých entit evidovaných systémem za určitým účelem. Přehled zaměstnanců (viz. Obrázek 2.1) nebo zákazníků bude obsahovat výstupní tabulku s patřičnými údaji o těchto entitách jako je jméno, příjmení, telefonní číslo, email, přihlašovací jméno aj. Další přehledy budou analogicky poskytovat data uložená v databázi pro vybrané entity.

Výstupy z modulu statistické analýzy budou obsahovat pouze relevantní data na základě zvoleného typu výstupu, tj. název statistické analýzy a výsledek pro provedení tohoto dotazu. Celý výstup je poté možno vygenerovat do kancelářského programu Microsoft Office Excel.



Obrázek 2.1: Přehled zaměstnanců

2.1.3 Funkce systému

Na další schůzce se zadavatelem byly upřesněny funkční požadavky a byly specifikovány některé nové funkční požadavky. Cílem bylo především detailně popsat funkce systému, procesy v organizaci a další omezení, které musí být v systému definovány. Při rozhovoru se zadavatelem jsem využil některé připravené otázky (vycházející z první schůzky) týkající se agendy HaPJP a také jsem nechal prostor zadavateli na jeho další připomínky a návrhy. Následujících několik odstavců bude popisovat získané poznatky.

Byly upřesněny základní funkce systému a také funkce, které by měly být realizovány v další fázi vývoje systému. Seznam funkcí je následující:

- Přihlášení do systému, odhlášení ze systému.
- Změna hesla, změna osobních údajů.
- Přehled, přidání, editace a smazání pořadů.
- Přidání, editace a smazání objednávky.
- Přehled objednávek v týdenním intervalu.
- Hledání objednávek.
- Práce s programy pro veřejnost (přehled, přidání, editace, smazání programu pro veřejnost).
- Přehled rezervací na program pro veřejnost.
- Přidání, editace a smazání rezervace.
- Provedení uzávěrky na další týden.
- Správa termínů (přehled, přidání, editace, smazání).
- Evidence zaměstnanců a zákazníků a funkce spojené s touto agendou.
- Přidání, editace a smazání textové poznámky u vybraného zákazníka (přehled poznámek).
- Přidání, editace a smazání aktuality (přehled aktualit).
- Přidání, editace a smazání služby u zaměstnance.
- Tisk týdenního rozpisu objednávek a programů pro veřejnost.
- Tisk přehledu objednávek.

- Tisk pokladní evidence.
- Přidání komentáře k vybranému pořadu, přehled komentářů (funkce pro zákazníky) a smazání komentáře (funkce pro zaměstnance).
- On-line objednání pořadu (funkce pro zákazníky).
- On-line provedení rezervace na program pro veřejnost (funkce pro zákazníky).
- Statistické výstupy z uložených dat.

Při přidávání nové objednávky do IS je stanoven minimální počet 20 osob na pořad, maximální počet je potom 100 osob. Minimální počet míst pro rezervaci není nijak omezen, maximální kapacita je stejná jako u objednávky.

IS by měl zobrazovat v přehledu objednávek počet volných nebo objednaných míst v rámci každé objednávky, což by mělo poskytovat snadnější orientaci při zadávání nové objednávky.

Systém bude obsahovat funkci Provedení uzávěrky na další týden. Jinými slovy to znamená, že na další týden už nebudou probíhat žádné změny. Funkci provádí zaměstnanec obvykle v pátek odpoledne. Administrátor systému bude mít možnost data změnit i po provedené uzávěrky pro případnou korekci hodnot apod.

Další funkční doplnění se týká posílání emailových zpráv. Pokud bude mít zákazník vyplněnou e-mailovou adresu, bude se generovat e-potvrzení objednávky, pokud k ní dojde.

Poslední částí IS z pohledu funkčních požadavků bude statistický modul, který bude nějakým způsobem vyhodnocovat data uložená v databázi. Na základě předem stanovených dotazů budou generovány výstupní informace, ze kterých by plynuly nějaké užitečné výsledky pro následné využití (např. cílená prezentace služeb apod.). Statistické výsledky by se vztahovaly vždy k nějakému časovému období (např. týdně, měsíčně, čtvrtletně nebo ročně).

Jako pomocný nástroj pro zadání funkcí systému byl zpracován tzv. seznam událostí a reakcí systému (viz. Tabulka 2.1), který má za úkol získat od zadavatele seznam všech požadovaných funkcí systému. Tento model obsahuje v našem případě pouze základní funkce navrhovaného IS.

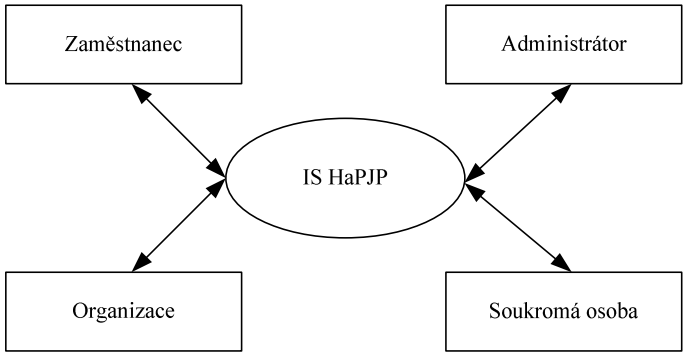
Událost	**Reakce**
Nový pořad	Zapiš nový záznam do systému
Nový program pro veřejnost	Zapiš nový program pro veřejnost do systému
Nová objednávka	Zapiš objednávku do systému
Změna objednávky	Zapiš změny do systému
Nová rezervace	Zapiš novou rezervaci do systému
Hledání objednávky	Vrať požadované záznamy podle zadaných kritérií
On-line objednávka	Zapiš novou objednávku do systému a pošli e-potvrzení
Provedení uzávěrky	Zapiš změny termínů ve vybraném týdnu

Nový komentář	Zapiš nový komentář do systému
Nepoužitelný termín	Zapiš záznam o termínu do systému
Tisk rozpisu	Vrať záznamy ze systému podle zadaných kritérií
Analýza dat	Vrať výsledky dotazu na základě kritérií a dat v systému

Tabulka 2.1: Seznam událostí a reakcí systému

2.1.4 Okolí systému

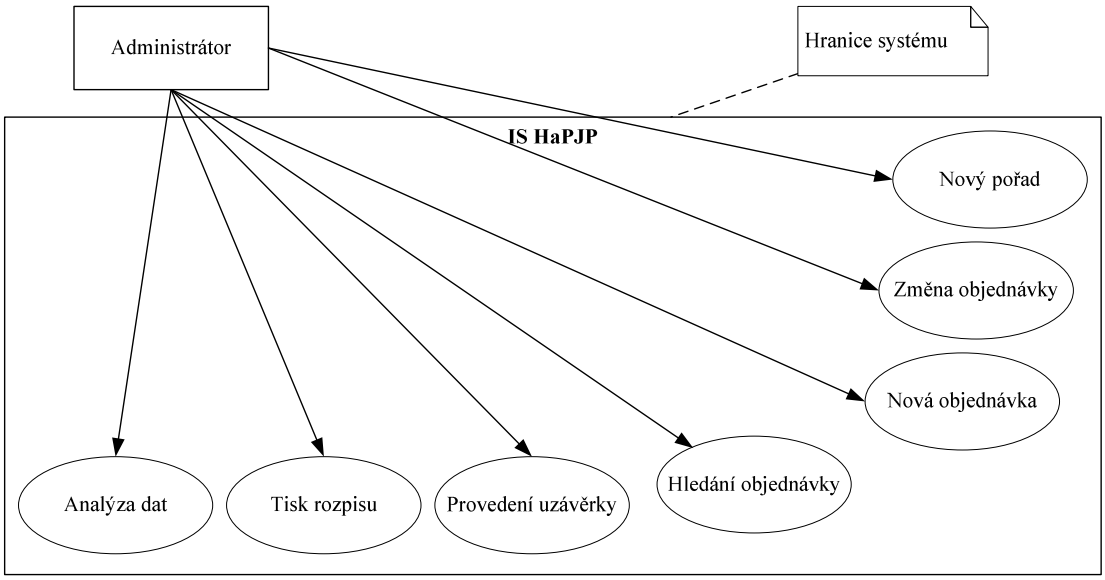
Je tvořeno všemi zaměstnanci HaPJP, externími spolupracovníky, kteří se podílejí na vstupních datech, dále pak jednotlivými zákazníky systému (organizace, soukromé osoby) a také uživateli Internetu, kteří se mohou dostat na týdenní rozpis pořadů a objednávek. Okolí systému názorně vyjadřuje kontextový diagram (viz. Obrázek 2.2).



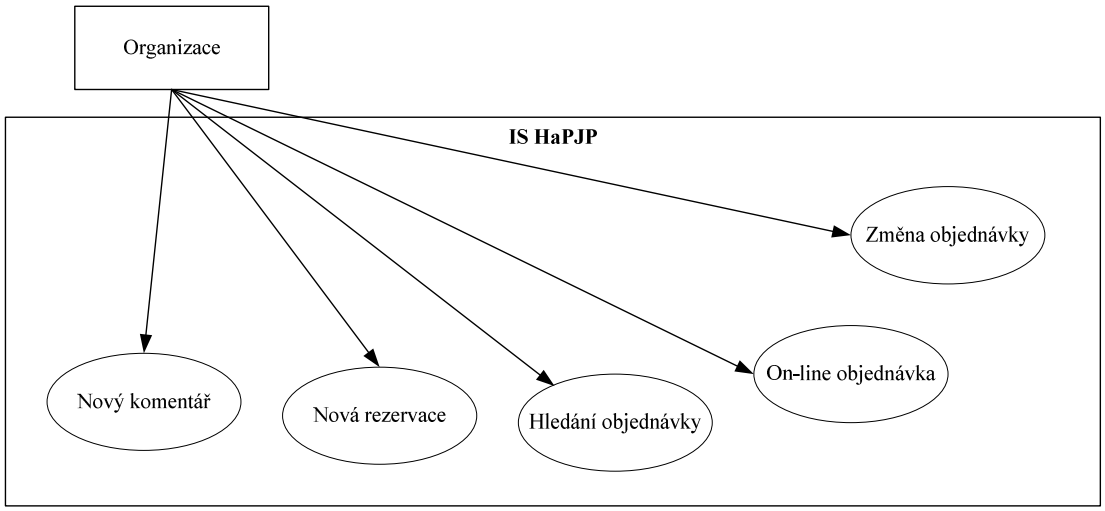
Obrázek 2.2: Kontextový diagram

Informační systém pro HaPJP budou podle specifikace požadavků využívat především 4 role, které jsou zobrazeny na výše uvedeném kontextovém diagramu. Jsou to role Administrátor, Zaměstnanec, Organizace a Soukromá osoba. Každá role bude mít k dispozici specifikovanou množinu funkcí. Tyto funkce budou dále popsány v kapitole Analýza (viz. 3) a také v kapitole Návrh implementace (viz. 4).

Na následujících obrázcích (viz. Obrázek 2.3, Obrázek 2.4) je vidět část modelu jednání pro řešený IS HaPJP. Tento model vlastně propojuje informace kontextového diagramu se seznamem událostí a reakcí systému.



Obrázek 2.3: Model jednání (Administrátor)



Obrázek 2.4: Model jednání (Organizace)

2.2 Nefunkční požadavky

Nefunkční požadavky na nově vytvářený informační systém HaPJP nebyly nějak detailně zadavatelem specifikovány, přesto se při běžném rozhovoru našlo několik vlastností nebo požadavků, které spadají do této kategorie. Seznam nefunkčních požadavků je následující:

- Nový IS by měl být dostupný ze všech pracovních stanic na HaPJP. Pokud k tomuto přidáme požadavek na on-line objednávku, potom se jako řešení nabízí webový informační systém. Při realizaci bude použito moderních IT technologií v této oblasti.
- Systém by měl mít snadné uživatelské ovládání a měl by odpovídat předem stanovenému grafickému formátu, který je definován v grafickém manuálu HaPJP (viz. Obsah přiloženého CD).
- Důležitým požadavkem, na který kladl zadavatel veliký důraz, je zálohování celého systému. Záloha by se měla provádět několikrát denně, aby se co nejvíce zamezilo případné ztrátě dat.
- Součástí realizace IS bude vypracování podrobné uživatelské dokumentace a také programátorské dokumentace, na kterou není kladen takový důraz jako na uživatelský manuál.
- Zadavatel také požaduje, aby byl nový systém bezpečný při běžném provozu a aby nedocházelo k jeho zneužití. Vzhledem k charakteru systému není potřeba nějakých zvláštních bezpečnostních opatření.
- Cenové náklady by měly být minimální. Systém by mohl být nasazen na serveru přímo na HaPJP, kde by bylo potřebné SW vybavení. Přítomnost serveru v síti VŠB-TU Ostrava zároveň vyřeší problémy s rychlostí a dostupností systému. Nepředpokládá se, že by se systémem pracovalo zároveň velké množství uživatelů, takže není kladen takový důraz na propustnost systému. Naopak by měl být systém spolehlivý pro běžné užití.

Nefunkční požadavky budou dále využity především při návrhu implementace, kde je třeba rozhodnout, jaké technologie se pro implementaci využijí, na jaké platformě systém poběží a další technické aspekty nutné k úspěšné realizaci IS. Z výše uvedených požadavků vyplývá, že se nejspíše bude jednat o webový informační systém, kde je třeba využít moderních technologií v této oblasti. Další detaily, týkající se návrhu implementace a dalších technických detailů souvisejících s tímto systémem, budou řešeny a popsány v dalších kapitolách, především potom v kapitole věnující se samostatnému návrhu implementace.

3. Analýza

Analýza je studium problému (jeho poznání, popis, modelování) dříve, než se začne s řešením. Úkolem analýzy je zpracování několika typů modelů budoucího systému. Model slouží jako podklad pro další řešení, je to abstraktní obraz budoucí reality (viz. [Šarm 1]).

Pro IS se vytváří 3 typy analýzy v tomto pořadí:

- Datová analýza
- Funkční analýza
- Dynamická analýza

Na základě provedené specifikace požadavků nyní zpracujeme 3 typy analýzy v pořadí uvedeném výše. Na začátku bude provedena datová analýza, která bude obsahovat konceptuální schéma databáze, tj. lineární zápis entit a jejich atributů, ERD diagram, datový slovník a další integritní omezení (dále jen IO). Potom provedeme funkční analýzu, tzn., že detailně popíšeme jednotlivé funkce systému a nakonec bude zpracována dynamická analýza, která vyjadřuje časovou následnost operací a jednotlivé stavy IS.

3.1 Datová analýza

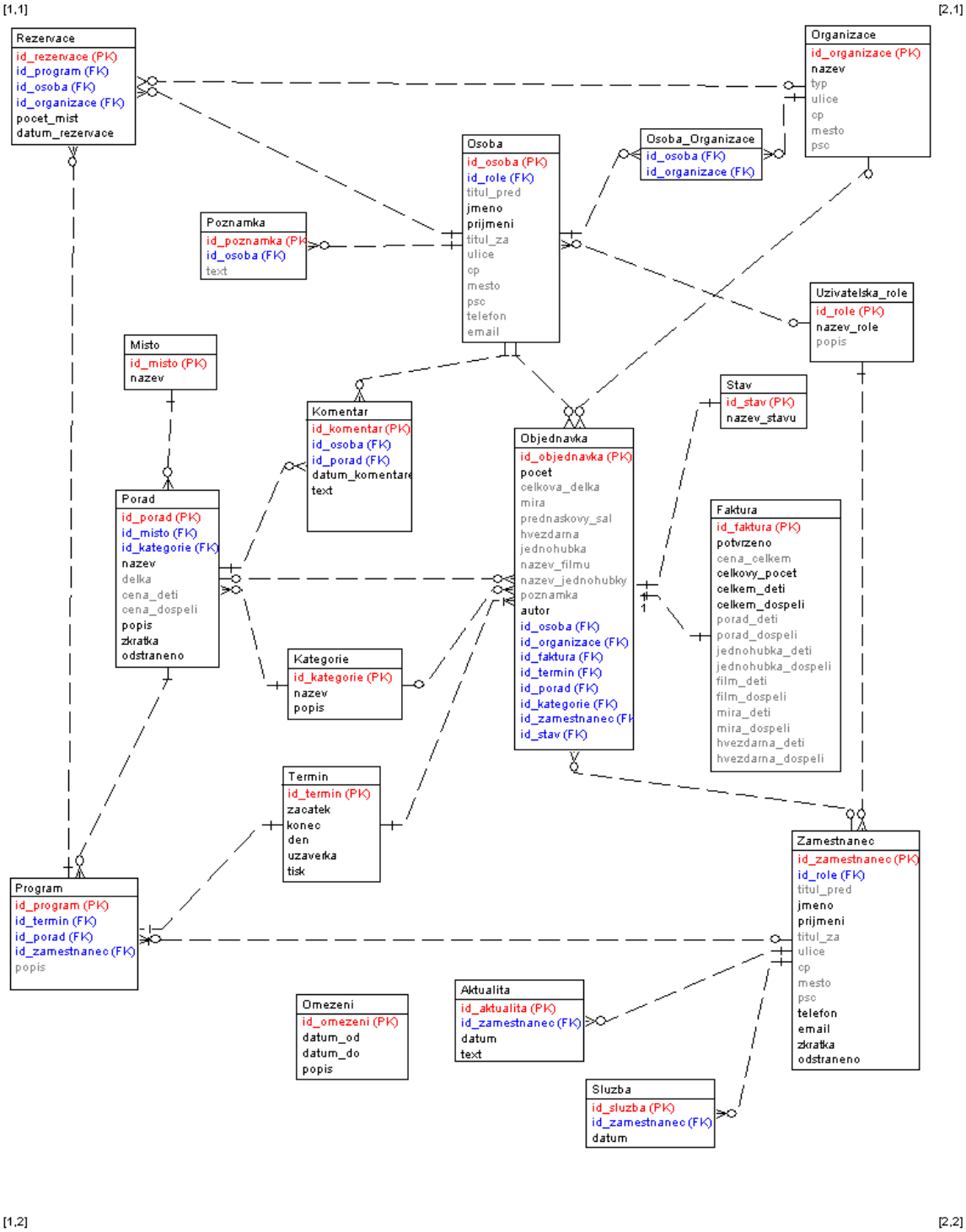
Datová analýza zpracovává datový model na konceptuální úrovni. Obsahem datové analýzy bude lineární zápis entit, ERD diagram a datový slovník obsahující další IO. Ze specifikace požadavků jsme provedli návrh struktury databáze. Primární klíče jsou podtrženy, cizí klíče jsou označeny kurzívou. Následující kapitola obsahuje výsledný lineární zápis.

3.1.1 Lineární zápis

- Aktualita (id_aktualita, *id_zamestnanec*, datum, text)
- Faktura (id_faktura, cena_celkem, celkovy_pocet, celkem_deti, celkem_dospeli, porad_deti, porad_dospeli, jednohubka_deti, jednohubka_dospeli, film_deti, film_dospeli, mira_deti, mira_dospeli, hvezdarna_deti, hvezdarna_dospeli)
- Kategorie (id_kategorie, nazev, popis) … číselník kategorií
- Komentar (id_komentar, *id_osoba*, *id_porad*, datum_komentare, text)
- Misto (id_misto, nazev) … umístění pořadu
- Objednavka (id_objednavka, *id_osoba*, *id_organizace*, *id_faktura*, *id_termin*, *id_porad*, *id_kategorie*, *id_zamestnanec*, *id_stav*, pocet, celkova_delka, mira, prednaskovy_sal, hvezdarna, jednohubka, nazev_filmu, nazev_jednohubky, poznamka, autor)
- Omezeni (id_omezeni, datum_od, datum_do, popis) … evidence nepovolených termínů
- Organizace (id_organizace, nazev, typ, ulice, cp, mesto, psc)

- Osoba (id_osoba, *id_role*, titul_pred, jmeno, prijmeni, titul_za, ulice, cp, mesto, psc, telefon, email) … evidence zákazníků
- Osoba_Organizace (*id_organizace*, *id_osoba*) … vazební tabulka, která vyjadřuje vazbu mezi organizací a konkrétní osobou
- Porad (id_porad, *id_misto*, *id_kategorie*, nazev, delka, cena_deti, cena_dospeli, popis, zkratka, odstraneno)
- Poznamka (id_poznamka, *id_osoba*, text) … evidence textových poznámek u osob
- Program (id_program, *id_termin*, *id_porad*, *id_zamestnanec*, popis) … evidence programů pro veřejnost
- Rezervace (id_rezervace, *id_program*, *id_osoba*, *id_organizace*, pocet_mist, datum_rezervace) … historie rezervací na programy pro veřejnost
- Sluzba (id_sluzba, *id_zamestnanec*, datum) … evidence služeb zaměstnanců
- Stav (id_stav, nazev_stavu) … číselník stavů objednávky
- Termin (id_termin, zacatek, konec, den, uzaverka, tisk) … evidence termínů pro objednávky a programy pro veřejnost
- Uzivatelska_role (id_role, nazev_role, popis) … číselník uživatelských rolí
- Zamestnanec (id_zamestnanec, *id_role*, titul_pred, jmeno, prijmeni, titul_za, ulice, cp, mesto, psc, telefon, email, zkratka, odstraneno) … evidence zaměstnanců

3.1.2 E-R (Entity-Relationship) Diagram



Obrázek 3.1: E-R (Entity-Relationship) Diagram

3.1.3 Datový slovník

Datový slovník uvedený v této kapitole (viz. Přílohy B) obsahuje popis jednotlivých tabulek a atributů, včetně detailních informací o těchto atributech. Datové typy jednotlivých atributů a délka jsou navrženy pro Microsoft SQL Server 2005 podle možností a doporučení pro tuto databázi (viz. [MS SQL 2005]). Datový slovník obsahuje také návrh indexů pro jednotlivé tabulky, tyto indexy byly navrženy v rámci kapitoly Návrh implementace (viz. 4).

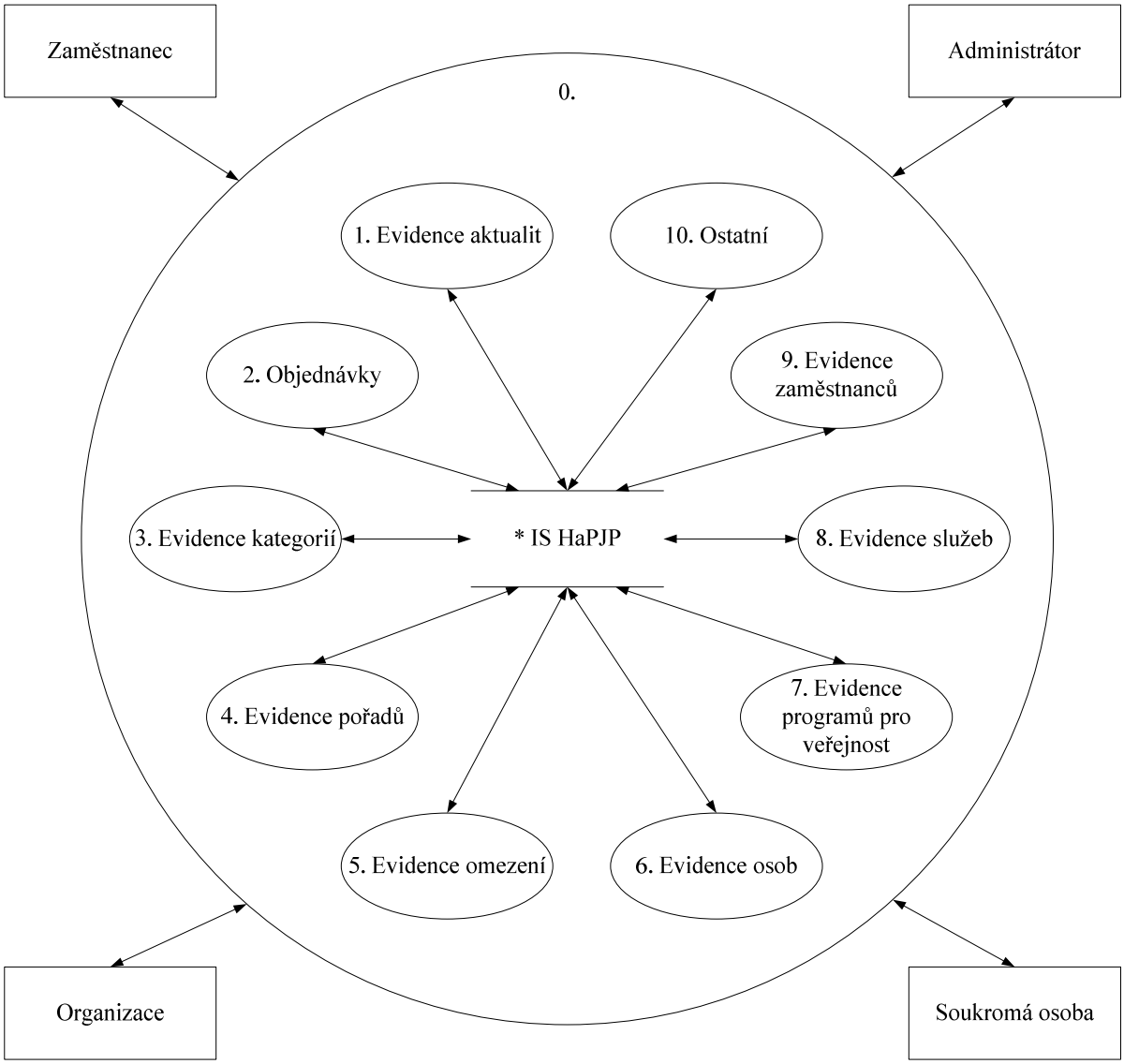
3.2 Funkční analýza

Funkční analýza vychází ze zadání IS, z požadovaných vstupů, výstupů a funkcí. Z těchto informací musíme nyní vytvořit funkční model systému. Funkční model bude obsahovat tyto 2 úrovně:

- vnější pohled (hrubý grafický náhled pomocí DFD)
- vnitřní pohled (podrobně rozpracovává algoritmy (minispecifikace) pro jednotlivé akce)

DFD (Data Flow Diagram), neboli diagram datových toků, je grafický nástroj pro modelování vztahů funkcí (algoritmů). Následující funkční analýza bude obsahovat hierarchickou strukturu DFD diagramů. Na vrcholu je zobrazen kontextový diagram (viz. Obrázek 2.2), následně bude zobrazen DFD úrovně 0 a potom DFD nejnižších úrovní, které budou popisovat elementární funkce systému. Elementární funkce budou zároveň obsahovat popis algoritmu (minispecifikaci) a také počáteční návrh formulářů pro danou funkci. Minispecifikace bude obsahovat také transakční analýzu, která byla provedena v rámci kapitoly Návrh implementace (viz. 4), ale pro přehlednost byly transakce doplněny do minispecifikací již nyní. V závěru této kapitoly bude rozšířen datový slovník (viz. 3.1.3) o datové toky použité v elementárních funkcích.

3.2.1 DFD diagram úrovně 0

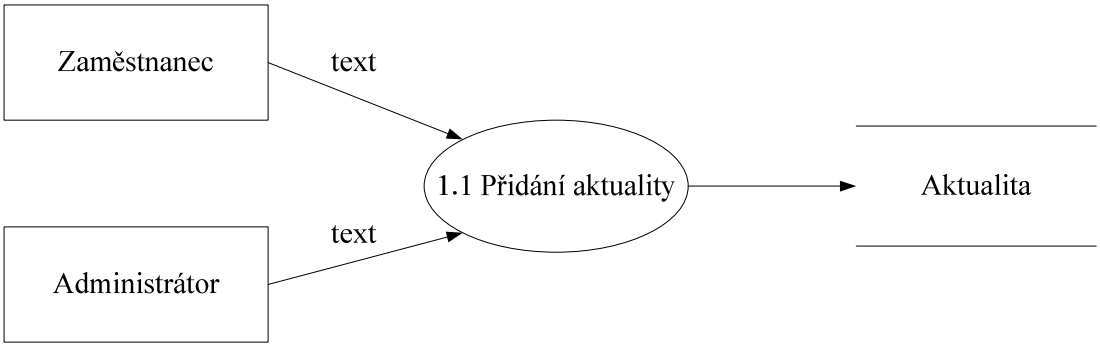


* IS HaPJP … jedná se o datovou paměť, která je zobecněním všech datových pamětí z datové analýzy

Obrázek 3.2: DFD diagram úrovně 0

3.2.2 DFD diagramy nejnižších úrovní + minispecifikace

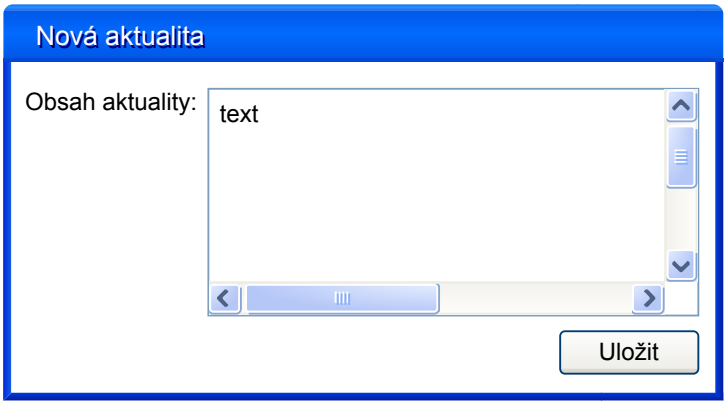
1.1 Přidání aktuality … funkce pro přidání nové aktuality do IS.



Obrázek 3.3: DFD (1.1 Přidání aktuality)

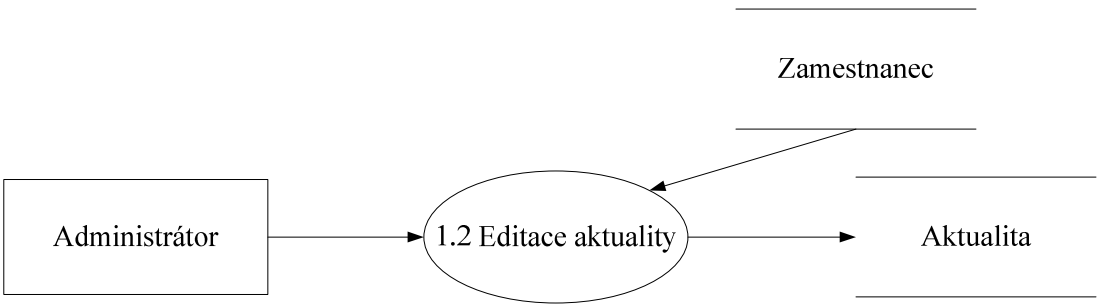
Algoritmus:

1. Zobraz formulář Nová aktualita



2. Uživatel vyplní formulář Nová aktualita
3. Ulož obsah aktuality z formuláře Nová aktualita do p.text
4. Zkontroluj integritní omezení. Jestliže je chyba, vypiš chybovou hlášku a jdi na bod 1
5. Ulož do p.id_zamestnanec hodnotu ID právě přihlášeného zaměstnance
6. Ulož do p.datum aktuální systémové datum
7. Zapiš p.id_zamestnanec, p.datum, p.text jako nový záznam do tabulky Aktualita
8. Jestliže se zápis neprovedl, vypiš chybovou hlášku a jdi na bod 1

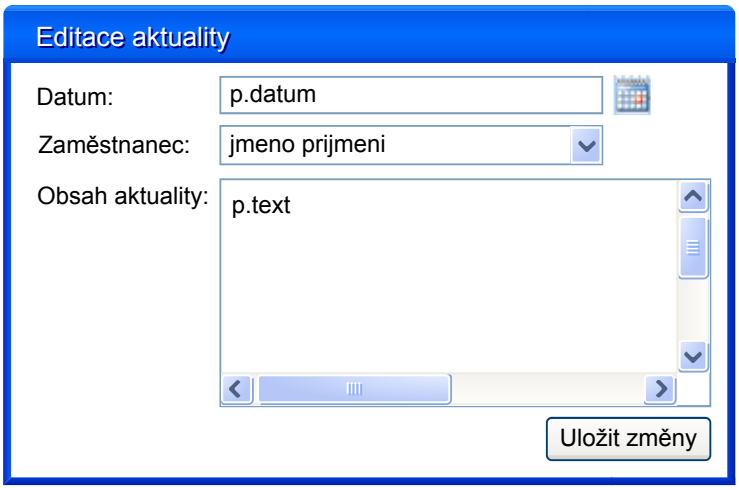
1.2 Editace aktuality … funkce pro editaci aktuality v databázi IS.



Obrázek 3.4: DFD (1.2 Editace aktuality)

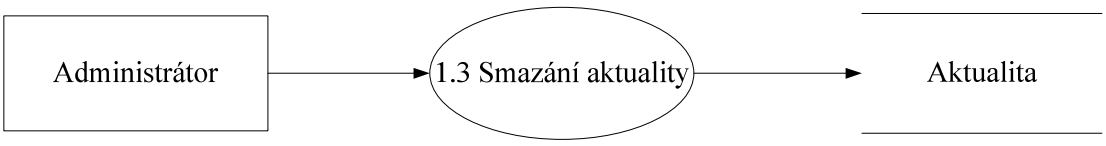
Algoritmus:

1. Zobraz formulář Seznam aktualit (viz. 1.4 Výpis aktualit)
2. Uživatel vybere aktualitu, kterou chce editovat
3. Ulož id_aktualita vybrané aktuality do p.id_aktualita
4. Vyber z tabulky Aktualita záznam, kde Aktualita.id_aktualita = p.id_aktualita a ulož hodnoty do p.id_zamestnanec, p.datum, p.text
5. Doplň aktualitu o atributy jmeno, prijmeni z tabulky Zamestnanec, kde Zamestnanec.id_zamestnanec = p.id_zamestnanec
6. Zobraz formulář Editace aktuality



7. Uživatel modifikuje hodnoty ve formuláři Editace aktuality
8. Ulož hodnoty z formuláře Editace aktuality do p.datum, p.text
9. Zkontroluj IO. Jestliže je chyba, vypiš chybovou hlášku a jdi na bod 6
10. Jestliže chce uživatel změnit ID zaměstnance
    PAK
    10.1. Zobraz seznam zaměstnanců z tabulky Zamestnanec
    10.2. Uživatel vybere zaměstnance
    10.3. Ulož id_zamestnanec vybraného zaměstnance do p.id_zamestnanec
11. Proveď zápis údajů p.id_zamestnanec, p.datum, p.text do tabulky Aktualita u záznamu, kde Aktualita.id_aktualita = p.id_aktualita
12. Jestliže se zápis neprovedl, vypiš chybovou hlášku a jdi na bod 1

1.3 Smazání aktuality … funkce, která odstraní aktualitu ze systému.



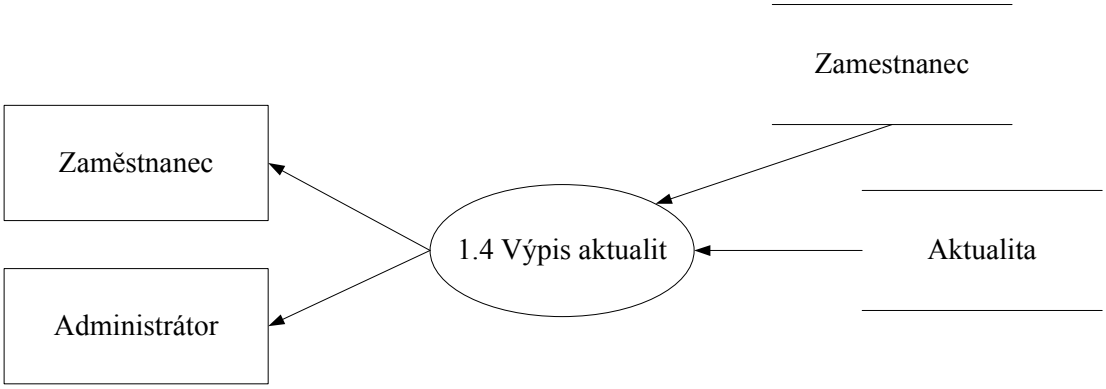
Obrázek 3.5: DFD (1.3 Smazání aktuality)

Algoritmus:

1. Zobraz formulář Seznam aktualit (viz. 1.4 Výpis aktualit)
2. Uživatel vybere aktualitu, kterou chce smazat, a klikne na tlačítko pro odstranění záznamu
3. Ulož id_aktualita vybrané aktuality do p.id_aktualita

4. Proveď odstranění záznamu z tabulky Aktualita, kde Aktualita.id_aktualita = p.id_aktualita
5. Jestliže se zápis neprovedl, vypiš chybovou hlášku a jdi na bod 1

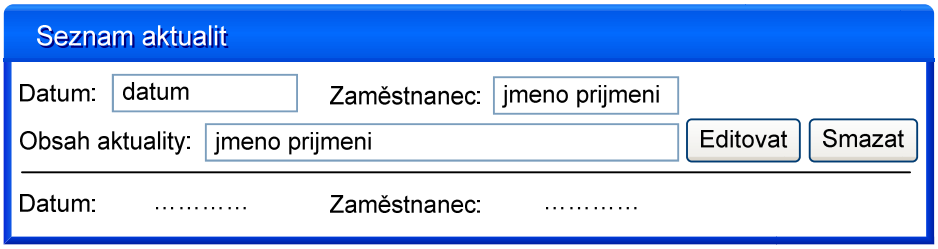
1.4 Výpis aktualit … funkce, která vypíše aktuality z databáze do formuláře Seznam aktualit.



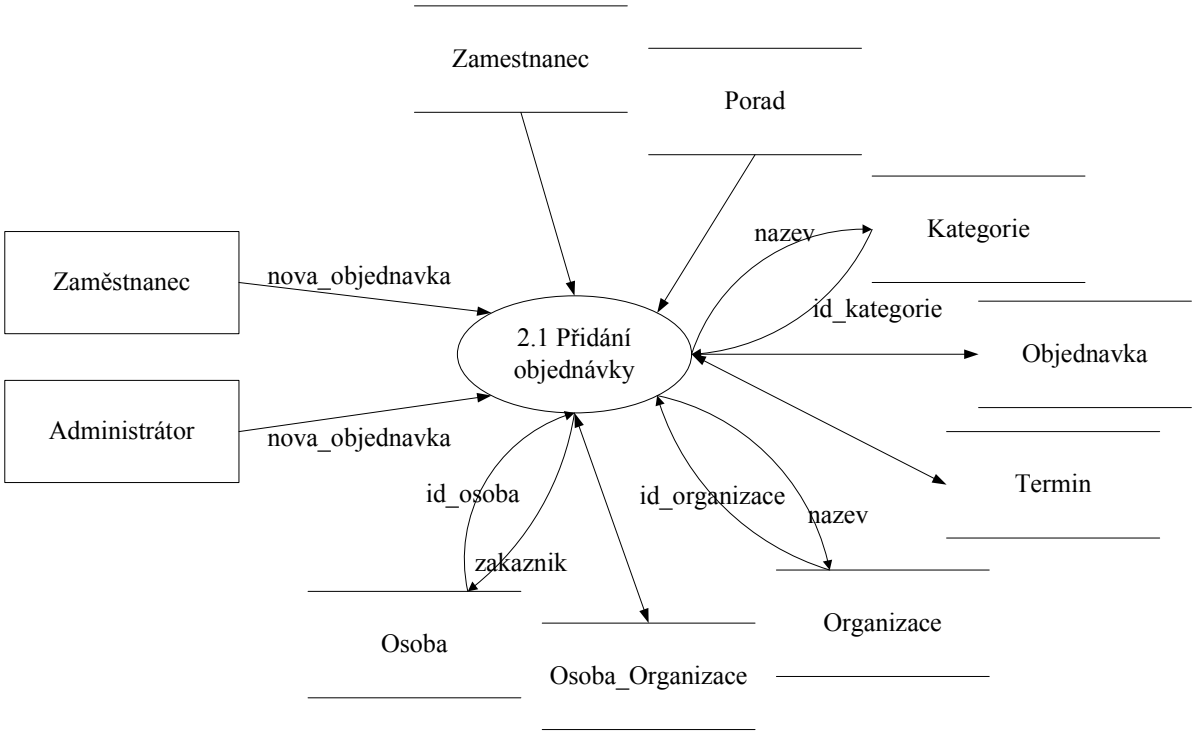
Obrázek 3.6: DFD (1.4 Výpis aktualit)

Algoritmus:

1. Načti z tabulky Aktualita všechny záznamy a setřiď je podle atributu datum
2. Doplň každou aktualitu o atributy jmeno, prijmeni z tabulky Zamestnanec, kde Zamestnanec.id_zamestnanec = Aktualita.id_zamestnanec
3. Vypiš záznamy ve formuláři Seznam aktualit podle daného setřídění včetně odkazů pro editaci a smazání aktuality



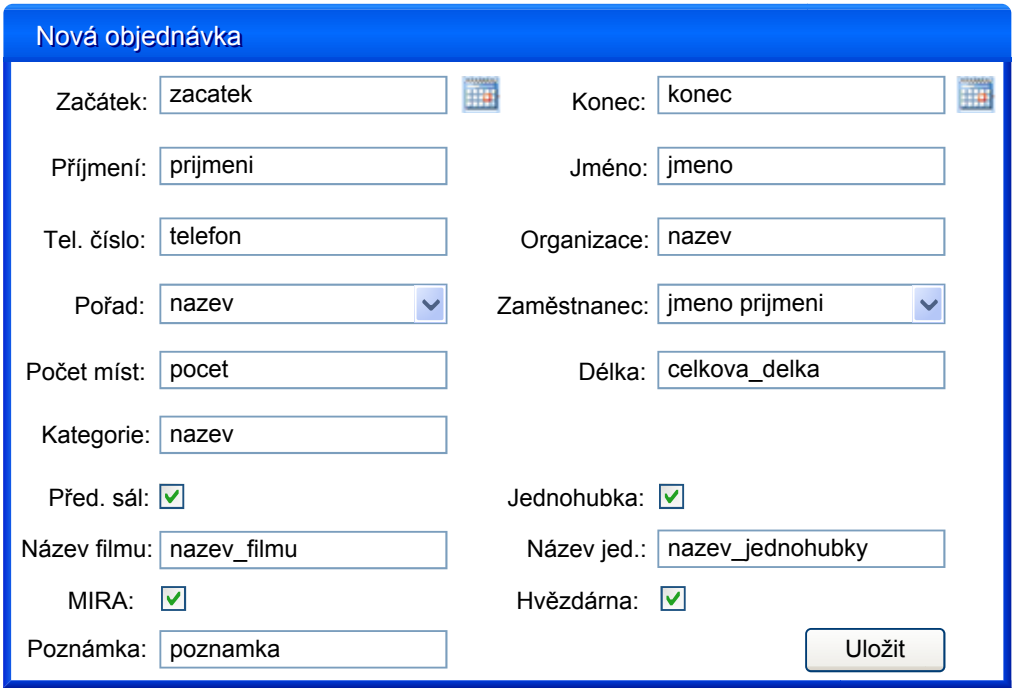
2.1 Přidání objednávky … funkce, která přidává novou objednávku do IS. Tuto funkci mohou v systému použít všechny role, ale minispecifikace bude odlišná podle toho, jestli jde o roli zaměstnance (Administrátor, Zaměstnanec) nebo zákazníka (Organizace, Soukromá osoba). Popíšu zde minispecifikaci z pohledu zaměstnanců HaPJP, neboť ta je komplexnější a popisuje více detailů. Přidání objednávky pro zákazníky systému probíhá v podstatě pouze tak, že uživatel specifikuje termín, potom si vybere pořad a počet míst. Provede se kontrola IO a pokud je vše v pořádku, tak je nový záznam uložen do databáze IS a vygeneruje se automaticky email, který přijde na emailovou adresu nastavenou v konfiguračním souboru (jako indikace o provedené změně). Princip indikace je zamýšlen pro on-line provoz celé aplikace.



Obrázek 3.7: DFD (2.1 Přidání objednávky)

Algoritmus:

1. Zobraz formulář Nová objednávka



2. Uživatel vyplní formulář Nová objednávka
3. Ulož hodnoty z formuláře Nová objednávka do p.zacatek, p.konec, p.prijmeni, p.jmeno, p.telefon, p.nazev_organizace, p.pocet, p.celkova_delka, p.nazev_kategorie,

p.prednaskovy_sal, p.jednohubka, p.nazev_filmu, p.nazev_jednohubky, p.mira, p.hvezdarna, p.poznamka

4. Zkontroluj integritní omezení. Jestliže je chyba, vypiš chybovou hlášku a jdi na bod 1
5. Jestliže chce uživatel změnit zaměstnance
   PAK
   5.1. Zobraz seznam zaměstnanců z tabulky Zamestnanec
   5.2. Uživatel vybere zaměstnance
   5.3. Ulož id_zamestnanec vybraného zaměstnance do p.id_zamestnanec
6. Jestliže chce uživatel změnit pořad
   PAK
   6.1. Zobraz seznam pořadů z tabulky Porad
   6.2. Uživatel vybere pořad
   6.3. Ulož id_porad vybraného pořadu do p.id_porad
7. Ulož do p.id_stav hodnotu id_stav z tabulky Stav, kde Stav.nazev_stavu = „Nová“
8. Ulož do p.autor hodnotu ID právě přihlášeného zaměstnance, který přidává novou objednávku do systému
9. Vyber Kategorie.id_kategorie z tabulky Kategorie, kde Kategorie.nazev = p.nazev_kategorie
10. Jestliže Kategorie.id_kategorie IS NOT NULL
    PAK
    10.1. Ulož atribut Kategorie.id_kategorie do p.id_kategorie
    JINAK
    10.2. Ulož p.nazev_kategorie do p.nazev
    10.3. Zapiš p.nazev do tabulky Kategorie jako nový záznam a vrať jeho id_kategorie do p.id_kategorie
11. BEGIN TRANSACTION
12. Vyber Organizace.id_organizace z tabulky Organizace, kde Organizace.nazev = p.nazev_organizace
13. Jestliže Organizace.id_organizace IS NOT NULL
    PAK
    13.1. Ulož atribut Organizace.id_organizace do p.id_organizace
    JINAK
    13.2. Ulož p.nazev_organizace do p.nazev
    13.3. Zapiš p.nazev do tabulky Organizace jako nový záznam a vrať jeho id_organizace do p.id_organizace
14. Vyber Osoba.id_osoba z tabulky Osoba, kde Osoba.jmeno = p.jmeno AND Osoba.prijmeni = p.prijmeni AND Osoba.telefon = p.telefon
15. Jestliže Osoba.id_osoba IS NOT NULL
    PAK
    15.1. Ulož Osoba.id_osoba do p.id_osoba
    JINAK
    15.2. Vyber Osoba.id_osoba z tabulky Osoba, kde Osoba.jmeno = p.jmeno AND Osoba.prijmeni = p.prijmeni

15.3. PRO KAŽDÉ Osoba.id_osoba DĚLEJ

15.3.1. Jestliže existuje záznam v tabulce Osoba_Organizace, kde Osoba_Organizace.id_osoba = Osoba.id_osoba AND Osoba_Organizace.id_organizace = p.id_organizace

PAK

15.3.1.1. Ulož Osoba.id_osoba do p.id_osoba

15.3.1.2. Zapiš p.telefon do tabulky Osoba, kde Osoba.id_osoba = p.id_osoba

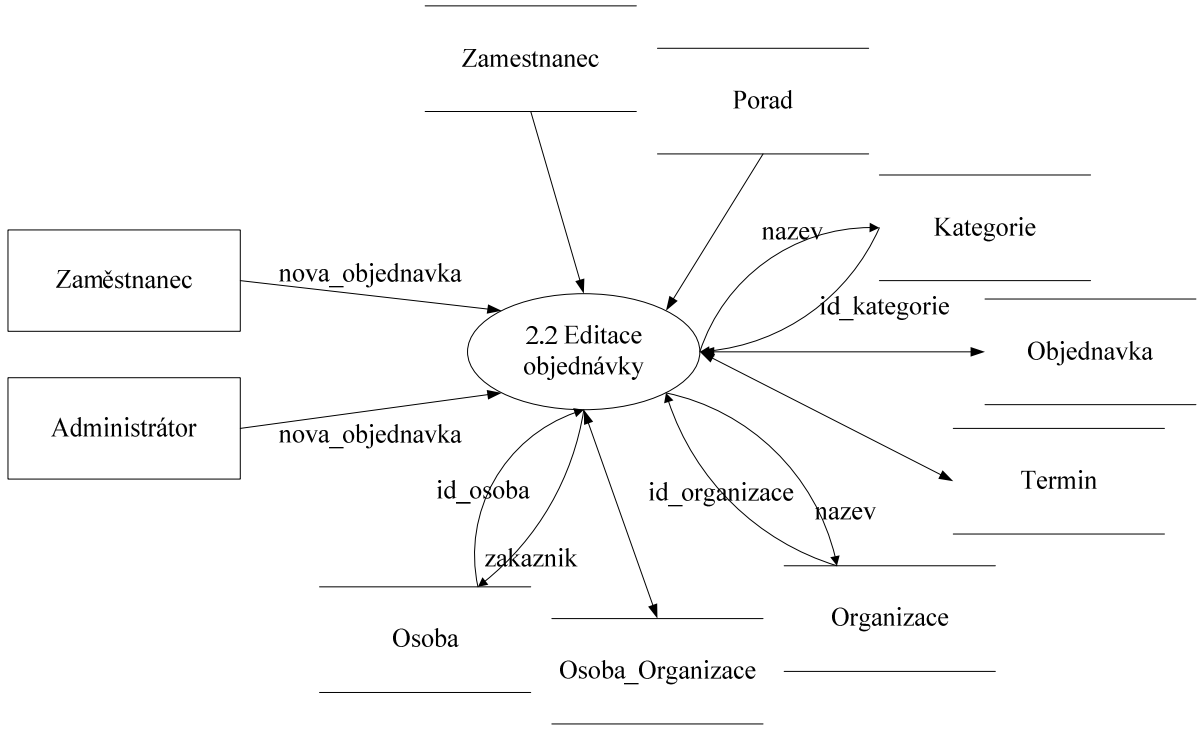
15.4. Jestliže neexistuje žádný záznam v tabulce Osoba, kde Osoba.jmeno = p.jmeno AND Osoba.prijmeni = p.prijmeni nebo p.id_osoba IS NULL

PAK

15.4.1. Zapiš p.jmeno, p.prijmeni, p.telefon do tabulky Osoba jako nový záznam a vrať jeho id_osoba do p.id_osoba

16. Jestliže neexistuje záznam v tabulce Osoba_Organizace, kde Osoba_Organizace.id_osoba = p.id_osoba AND Osoba_Organizace.id_organizace = p.id_organizace
    PAK
    16.1. Zapiš p.id_organizace, p.id_osoba do tabulky Osoba_Organizace jako nový záznam
17. END TRANSACTION
18. Jestliže se zápis neprovedl, vypiš chybovou hlášku a jdi na bod 1
19. BEGIN TRANSACTION
20. Vyber Termin.id_termin z tabulky Termin, kde Termin.zacatek = p.zacatek AND Termin.konec = p.konec
21. Jestliže Termin.id_termin IS NOT NULL
    PAK
    21.1. Ulož Termin.id_termin do p.id_termin
    JINAK
    21.2. Ulož hodnotu názvu dne v týdnu získané z hodnoty p.zacatek do p.den
    21.3. Zapiš p.zacatek, p.konec, p.den, FALSE, FALSE do tabulky Termin jako nový záznam a vrať jeho id_termin do p.id_termin
22. Zapiš do tabulky Faktura nový záznam a vrať jeho id_faktura do p.id_faktura
23. Zapiš p.id_osoba, p.id_organizace, p.id_faktura, p.id_termin, p.id_porad, p.id_kategorie, p.id_zamestnanec, p.id_stav, p.pocet, p.celkova_delka, p.mira, p.prednaskovy_sal, p.jednohubka, p.hvezdarna, p.nazev_filmu, p.nazev_jednohubky, p.poznamka, p.autor do tabulky Objednavka jako nový záznam
24. END TRANSACTION
25. Jestliže se zápis neprovedl, vypiš chybovou hlášku a jdi na bod 1

2.2 Editace objednávky … funkce pro editaci objednávky v databázi IS. Opět jsou zde odlišnosti mezi rolemi systému. Zaměstnanci mají komplexnější možnosti, proto je zde popsána minispecifikace z pohledu zaměstnanců HaPJP. Zákazníci mají možnost editovat pouze ty informace, které mohou samostatně zadávat (viz. 2.1 Přidání objednávky). Princip indikace změny objednávky pomocí emailové zprávy je zde stejný. Ještě to bude popsáno v kapitole Dynamická analýza (viz. 3.3).

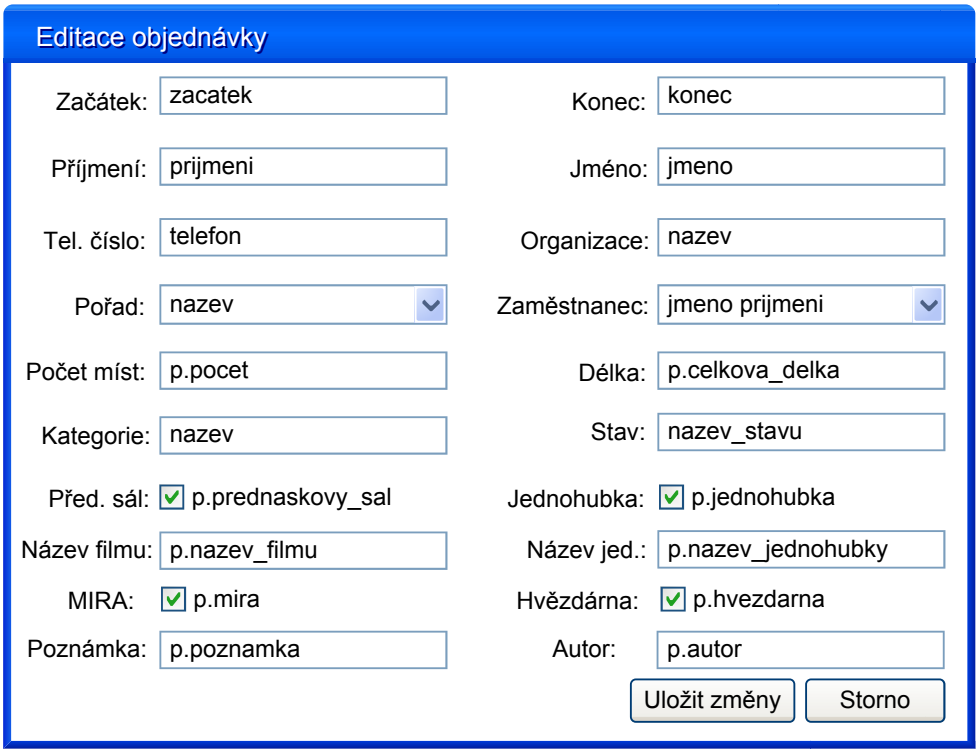


Obrázek 3.8: DFD (2.2 Editace objednávky)

Algoritmus:

1. Zobraz formulář Objednávka (viz. 2.5 Výpis objednávky) s vybranou objednávkou, kterou chce uživatel editovat
2. Ulož id_objednavka vybrané objednávky do p.id_objednavka
3. Uživatel klikne na tlačítko pro editaci objednávky
4. Načti z tabulky Objednavka záznam, kde Objednavka.id_objednavka = p.id_objednavka a ulož atributy do p.id_osoba, p.id_organizace, p.id_termin, p.id_porad, p.id_kategorie, p.id_zamestnanec, p.id_stav, p.pocet, p.celkova_delka, p.prednaskovy_sal, p.jednohubka, p.mira, p.hvezdarna, p.nazev_filmu, p.nazev_jednohubky, p.poznamka, p.autor
5. Doplň objednávku o atributy jmeno, prijmeni z tabulky Zamestnanec, kde Zamestnanec.id_zamestnanec = p.id_zamestnanec
6. Doplň objednávku o atributy nazev z tabulky Porad, kde Porad.id_porad = p.id_porad
7. Doplň objednávku o atributy nazev z tabulky Kategorie, kde Kategorie.id_kategorie = p.id_kategorie
8. Doplň objednávku o atributy nazev z tabulky Organizace, kde Organizace.id_organizace = p.id_organizace
9. Doplň objednávku o atributy zacatek, konec z tabulky Termin, kde Termin.id_termin = p.id_termin
10. Doplň objednávku o atributy nazev_stavu z tabulky Stav, kde Stav.id_stav = p.id_stav
11. Doplň objednávku o atributy jmeni, prijmeni, telefon z tabulky Osoba, kde Osoba.id_osoba = p.id_osoba

12. Vypiš záznam doplněný o výše uvedené atributy ve formuláři Editace objednávky



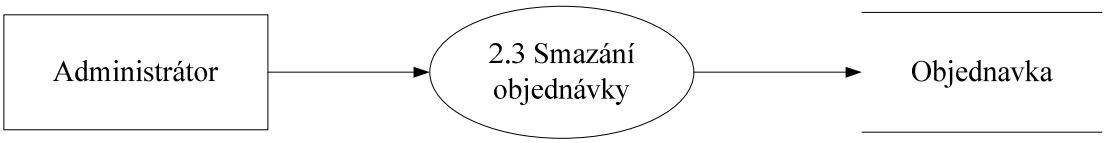
13. Uživatel modifikuje hodnoty ve formuláři Editace objednávky
14. Ulož hodnoty z formuláře Editace objednávky do p.zacatek, p.konec, p.prijmeni, p.jmeno, p.telefon, p.nazev_organizace, p.pocet, p.celkova_delka, p.nazev_kategorie, p.prednaskovy_sal, p.jednohubka, p.nazev_filmu, p.nazev_jednohubky, p.mira, p.hvezdarna, p.poznamka
15. Zkontroluj integritní omezení. Jestliže je chyba, vypiš chybovou hlášku a jdi na bod 1
16. Jestliže chce uživatel změnit zaměstnance
    PAK
    16.1. Zobraz seznam zaměstnanců z tabulky Zamestnanec
    16.2. Uživatel vybere zaměstnance
    16.3. Ulož id_zamestnanec vybraného zaměstnance do p.id_zamestnanec
17. Jestliže chce uživatel změnit pořad
    PAK
    17.1. Zobraz seznam pořadů z tabulky Porad
    17.2. Uživatel vybere pořad
    17.3. Ulož id_porad vybraného pořadu do p.id_porad
18. Ulož do p.id_stav hodnotu id_stav z tabulky Stav, kde Stav.nazev_stavu = „Upravená“
19. Ulož do p.autor hodnotu ID právě přihlášeného zaměstnance, který edituje objednávku v systému
20. Vyber Kategorie.id_kategorie z tabulky Kategorie, kde Kategorie.nazev = p.nazev_kategorie
21. Jestliže Kategorie.id_kategorie IS NOT NULL
    PAK

21.1. Ulož atribut Kategorie.id_kategorie do p.id_kategorie
JINAK
21.2. Ulož p.nazev_kategorie do p.nazev
21.3. Zapiš p.nazev do tabulky Kategorie jako nový záznam a vrať jeho id_kategorie do p.id_kategorie

22. BEGIN TRANSACTION
23. Vyber Organizace.id_organizace z tabulky Organizace, kde Organizace.nazev = p.nazev_organizace
24. Jestliže Organizace.id_organizace IS NOT NULL
PAK
24.1. Ulož atribut Organizace.id_organizace do p.id_organizace
JINAK
24.2. Ulož p.nazev_organizace do p.nazev
24.3. Zapiš p.nazev do tabulky Organizace jako nový záznam a vrať jeho id_organizace do p.id_organizace
25. Vyber Osoba.id_osoba z tabulky Osoba, kde Osoba.jmeno = p.jmeno AND Osoba.prijmeni = p.prijmeni AND Osoba.telefon = p.telefon
26. Jestliže Osoba.id_osoba IS NOT NULL
PAK
26.1. Ulož Osoba.id_osoba do p.id_osoba
JINAK
26.2. Vyber Osoba.id_osoba z tabulky Osoba, kde Osoba.jmeno = p.jmeno AND Osoba.prijmeni = p.prijmeni
26.3. PRO KAŽDÉ Osoba.id_osoba DĚLEJ
26.3.1. Jestliže existuje záznam v tabulce Osoba_Organizace, kde Osoba_Organizace.id_osoba = Osoba.id_osoba AND Osoba_Organizace.id_organizace = p.id_organizace
PAK
26.3.1.1. Ulož Osoba.id_osoba do p.id_osoba
26.3.1.2. Zapiš p.telefon do tabulky Osoba, kde Osoba.id_osoba = p.id_osoba
26.4. Jestliže neexistuje žádný záznam v tabulce Osoba, kde Osoba.jmeno = p.jmeno AND Osoba.prijmeni = p.prijmeni nebo p.id_osoba IS NULL
PAK
26.4.1. Zapiš p.jmeno, p.prijmeni, p.telefon do tabulky Osoba jako nový záznam a vrať jeho id_osoba do p.id_osoba
27. Jestliže neexistuje záznam v tabulce Osoba_Organizace, kde Osoba_Organizace.id_osoba = p.id_osoba AND Osoba_Organizace.id_organizace = p.id_organizace
PAK
27.1. Zapiš p.id_organizace, p.id_osoba do tabulky Osoba_Organizace jako nový záznam
28. END TRANSACTION
29. Jestliže se zápis neprovedl, vypiš chybovou hlášku a jdi na bod 12
30. BEGIN TRANSACTION

31. Vyber Termin.id_termin z tabulky Termin, kde Termin.zacatek = p.zacatek AND Termin.konec = p.konec
32. Jestliže Termin.id_termin IS NOT NULL
    PAK
    32.1. Ulož Termin.id_termin do p.id_termin
    JINAK
    32.2. Ulož hodnotu názvu dne v týdnu získané z hodnoty p.zacatek do p.den
    32.3. Proveď zápis údajů p.zacatek, p.konec, p.den do tabulky Termin, kde Termin.id_termin = p.id_termin a vrať jeho id_termin do p.id_termin
33. Ulož p.pocet do p.celkovy_pocet
34. Proveď zápis údaje p.celkovy_pocet do tabulky Faktura, kde Faktura.id_faktura = p.id_faktura
35. Proveď zápis údajů p.id_osoba, p.id_organizace, p.id_faktura, p.id_termin, p.id_porad, p.id_kategorie, p.id_zamestnanec, p.id_stav, p.pocet, p.celkova_delka, p.mira, p.prednaskovy_sal, p.jednohubka, p.hvezdarna, p.nazev_filmu, p.nazev_jednohubky, p.poznamka, p.autor do tabulky Objednavka, kde Objednavka.id_objednavka = p.id_objednavka
36. END TRANSACTION
37. Jestliže se zápis neprovedl, vypiš chybovou hlášku a jdi na bod 12
38. Jestliže selhalo cokoli jiného, vypiš chybovou hlášku a jdi na bod 1

2.3 Smazání objednávky … funkce, která odstraní objednávku z databáze IS.



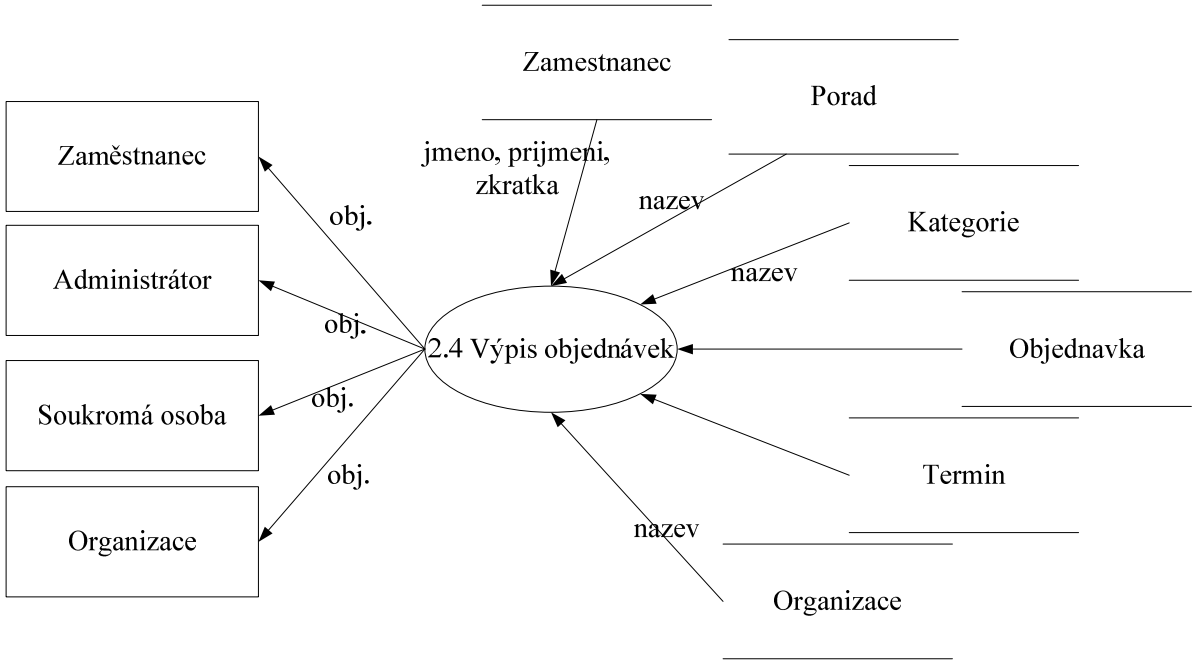
Obrázek 3.9: DFD (2.3 Smazání objednávky)

Algoritmus:

1. Zobraz formulář Objednávka (viz. 2.5 Výpis objednávky) s vybranou objednávkou, kterou chce uživatel smazat
2. Ulož id_objednavka vybrané objednávky do p.id_objednavka a id_faktura do p.id_faktura a id_termin do p.id_termin
3. Uživatel klikne na tlačítko pro odstranění objednávky ze systému
4. BEGIN TRANSACTION
5. Jestliže neexistuje záznam v tabulce Objednavka, kde Objednavka.id_termin = p.id_termin AND Objednavka.id_objednavka != p.id_objednavka
   PAK
   5.1. Proveď odstranění záznamu z tabulky Termin, kde Termin.id_termin = p.id_termin
6. Proveď odstranění záznamu z tabulky Faktura, kde Faktura.id_faktura = p.id_faktura
7. Proveď odstranění záznamu z tabulky Objednavka, kde Objednavka.id_objednavka = p.id_objednavka
8. END TRANSACTION

9. Jestliže se zápis neprovedl, vypiš chybovou hlášku a jdi na bod 1

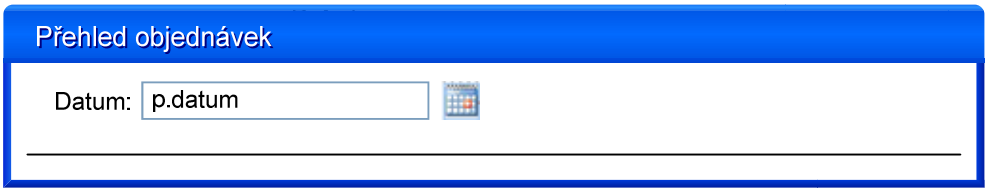
2.4 Výpis objednávek … funkce, která vypíše přehled objednávek v zadaném týdnu.



Obrázek 3.10: DFD (2.4 Výpis objednávek)

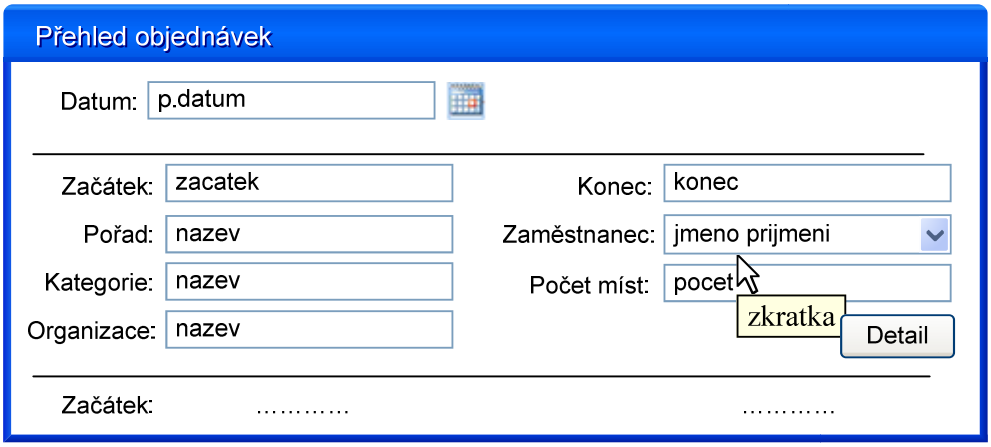
Algoritmus:

1. Zobraz záhlaví formuláře Přehled objednávek obsahující hodnotu p.datum

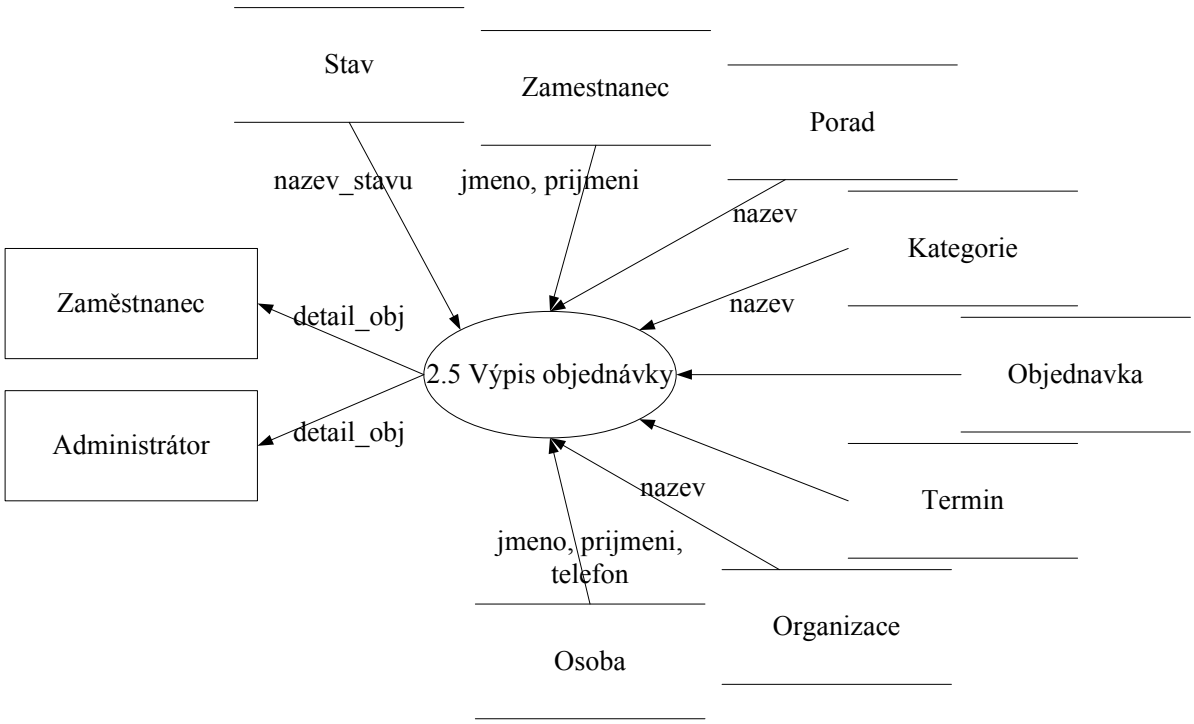


2. Ulož do p.datum hodnotu aktuálního systémového data
3. Jestliže chce uživatel změnit hodnotu p.datum
   PAK
   3.1. Uživatel zadá do vstupního pole novou hodnotu položky datum (dd.MM.yyyy)
   3.2. Ulož datum do p.datum
4. Z hodnoty p.datum zjisti počátek týdne a konec týdne a ulož do p.zacatek a p.konec
5. Ulož do p.id_stav hodnotu id_stav z tabulky Stav, kde Stav.nazev_stavu = „Stornovaná“
6. Načti z tabulky Objednavka všechny záznamy, kde Termin.zacatek <= p.zacatek AND Termin.konec <= p.konec AND Termin.id_termin = Objednavka.id_termin AND Objednavka.id_stav != p.id_stav a setřiď podle atributu Termin.zacatek
7. Doplň každou objednávku o atributy jmeno, prijmeni, zkratka z tabulky Zamestnanec, kde Zamestnanec.id_zamestnanec = Objednavka.id_zamestnanec
8. Doplň každou objednávku o atributy nazev z tabulky Porad, kde Porad.id_porad = Objednavka.id_porad

9. Doplň každou objednávku o atributy nazev z tabulky Kategorie, kde Kategorie.id_kategorie = Objednavka.id_kategorie
10. Doplň každou objednávku o atributy nazev z tabulky Organizace, kde Organizace.id_organizace = Objednavka.id_organizace
11. Doplň každou objednávku o atributy zacatek, konec z tabulky Termin, kde Termin.id_termin = Objednavka.id_termin
12. Vypiš záznamy doplněné o výše uvedené atributy ve formuláři Přehled objednávek podle daného setřídění včetně odkazů pro detail objednávky



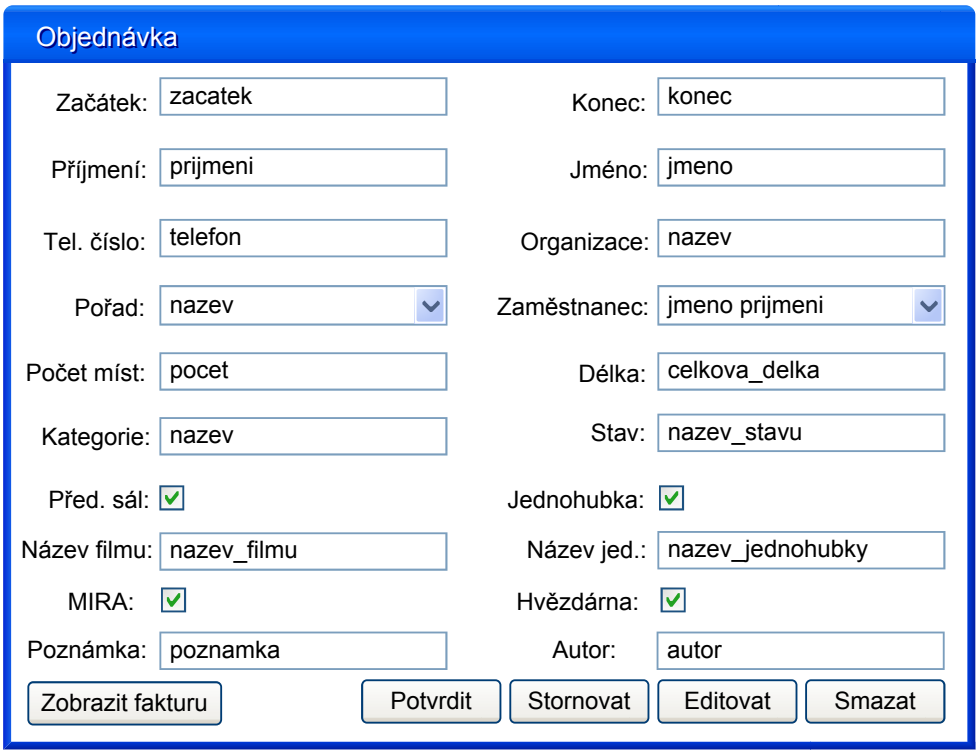
2.5 Výpis objednávky … funkce, která poskytuje detailní výpis vybrané objednávky včetně odkazů na další funkce spojené s vybranou objednávkou.



Obrázek 3.11: DFD (2.5 Výpis objednávky)

Algoritmus:

1. Zobraz formulář Přehled objednávek (viz. 2.4 Výpis objednávek)
2. Uživatel vybere objednávku, kterou chce detailně vypsat
3. Ulož id_objednavka vybrané objednávky do p.id_objednavka
4. Načti z tabulky Objednavka záznam, kde Objednavka.id_objednavka = p.id_objednavka a ulož atributy do p.id_osoba, p.id_organizace, p.id_termin, p.id_porad, p.id_kategorie, p.id_zamestnanec, p.id_stav, p.pocet, p.celkova_delka, p.prednaskovy_sal, p.jednohubka, p.mira, p.hvezdarna, p.nazev_filmu, p.nazev_jednohubky, p.poznamka, p.autor
5. Doplň objednávku o atributy jmeno, prijmeni z tabulky Zamestnanec, kde Zamestnanec.id_zamestnanec = p.id_zamestnanec
6. Doplň objednávku o atributy nazev z tabulky Porad, kde Porad.id_porad = p.id_porad
7. Doplň objednávku o atributy nazev z tabulky Kategorie, kde Kategorie.id_kategorie = p.id_kategorie
8. Doplň objednávku o atributy nazev z tabulky Organizace, kde Organizace.id_organizace = p.id_organizace
9. Doplň objednávku o atributy zacatek, konec z tabulky Termin, kde Termin.id_termin = p.id_termin
10. Doplň objednávku o atributy nazev_stavu z tabulky Stav, kde Stav.id_stav = p.id_stav
11. Doplň objednávku o atributy jmeni, prijmeni, telefon z tabulky Osoba, kde Osoba.id_osoba = p.id_osoba
12. Vypiš záznam doplněný o výše uvedené atributy ve formuláři Objednávka včetně odkazů pro další funkce spojené s vybranou objednávkou



2.6 Potvrzení objednávky … funkce, která provádí potvrzení objednávky, tj. změní stav vybrané objednávky na hodnotu „Potvrzená“. Změny stavu objednávky jsou popsány v kapitole Dynamická analýza (viz. 3.3).

Obrázek 3.12: DFD (2.6 Potvrzení objednávky)

Algoritmus:

1. Zobraz formulář Objednávka (viz. 2.5 Výpis objednávky) s vybranou objednávkou, kterou chce uživatel potvrdit
2. Ulož id_objednavka vybrané objednávky do p.id_objednavka
3. Uživatel klikne na tlačítko pro potvrzení objednávky v systému
4. Ulož do p.id_stav atribut id_stav z tabulky Stav, kde Stav.nazev_stavu = „Potvrzená“
5. Proveď zápis údaje p.id_stav do tabulky Objednavka, kde Objednavka.id_objednavka = p.id_objednavka
6. Jestliže se zápis neprovedl, vypiš chybovou hlášku a jdi na bod 1

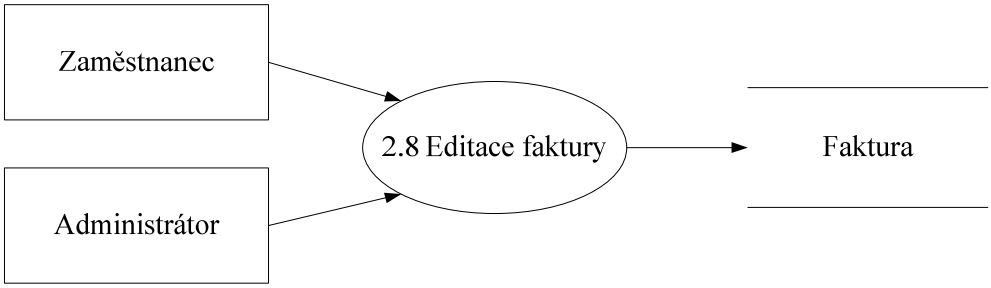
2.7 Stornování objednávky … funkce, která provádí stornování objednávky, tj. změní stav vybrané objednávky na hodnotu „Stornovaná“. Změny stavu objednávky jsou popsány v kapitole Dynamická analýza (viz. 3.3).

Obrázek 3.13: DFD (2.7 Stornování objednávky)

Algoritmus:

1. Zobraz formulář Objednávka (viz. 2.5 Výpis objednávky) s vybranou objednávkou, kterou chce uživatel stornovat
2. Ulož id_objednavka vybrané objednávky do p.id_objednavka
3. Uživatel klikne na tlačítko pro stornování objednávky v systému
4. Ulož do p.id_stav atribut id_stav z tabulky Stav, kde Stav.nazev_stavu = „Stornovaná“
5. Proveď zápis údaje p.id_stav do tabulky Objednavka, kde Objednavka.id_objednavka = p.id_objednavka
6. Jestliže se zápis neprovedl, vypiš chybovou hlášku a jdi na bod 1

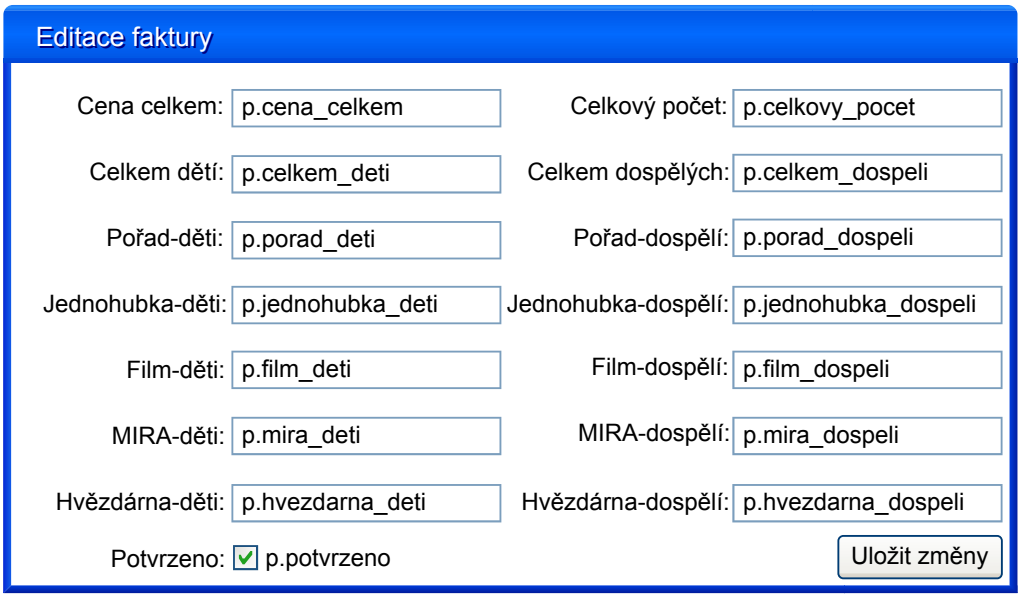
2.8 Editace faktury … funkce, která poskytuje editaci vybrané faktury v databázi IS.



Obrázek 3.14: DFD (2.8 Editace faktury)

Algoritmus:

1. Zobraz formulář Faktura (viz. 2.9 Výpis faktury)
2. Ulož id_faktura zobrazené faktury do p.id_faktura
3. Uživatel klikne na tlačítkou pro editaci faktury
4. Načti z tabulky Faktura záznam, kde Faktura.id_faktura = p.id_faktura a ulož atributy do p.potvrzeno, p.cena_celkem, p.celkovy_pocet, p.celkem_deti, p.celkem_dospeli, p.porad_deti, p.porad_dospeli, p.jednohubka_deti, p.jednohubka_dospeli, p.film_deti, p.film_dospeli, p.mira_deti, p.mira_dospeli, p.hvezdarna_deti, p.hvezdarna_dospeli
5. Zobraz formulář Editace faktury

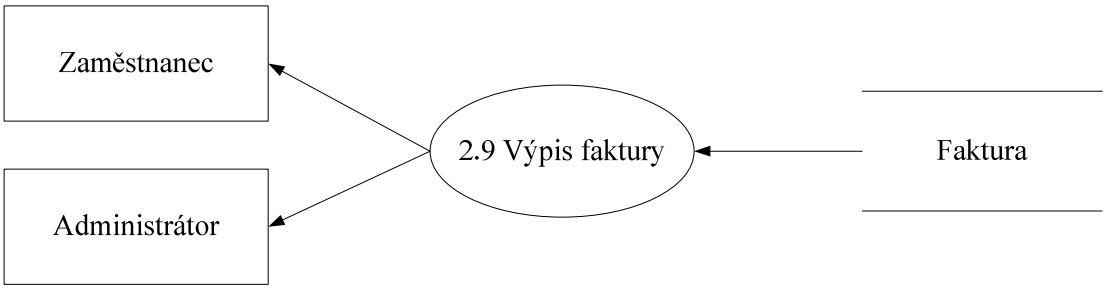


6. Uživatel modifikuje hodnoty ve formuláři Editace faktury
7. Ulož hodnoty z formuláře Editace faktury do p.potvrzeno, p.cena_celkem, p.celkovy_pocet, p.celkem_deti, p.celkem_dospeli, p.porad_deti, p.porad_dospeli, p.jednohubka_deti, p.jednohubka_dospeli, p.film_deti, p.film_dospeli, p.mira_deti, p.mira_dospeli, p.hvezdarna_deti, p.hvezdarna_dospeli
8. Zkontroluj IO. Jestliže je chyba, vypiš chybovou hlášku a jdi na bod 5.
9. Proveď zápis údajů p.potvrzeno, p.cena_celkem, p.celkovy_pocet, p.celkem_deti, p.celkem_dospeli, p.porad_deti, p.porad_dospeli, p.jednohubka_deti, p.jednohubka_dospeli,

p.film_deti, p.film_dospeli, p.mira_deti, p.mira_dospeli, p.hvezdarna_deti, p.hvezdarna_dospeli do tabulky Faktura u záznamu, kde Faktura.id_faktura = p.id_faktura

10. Jestliže se zápis neprovedl, vypiš chybovou hlášku a jdi na bod 1

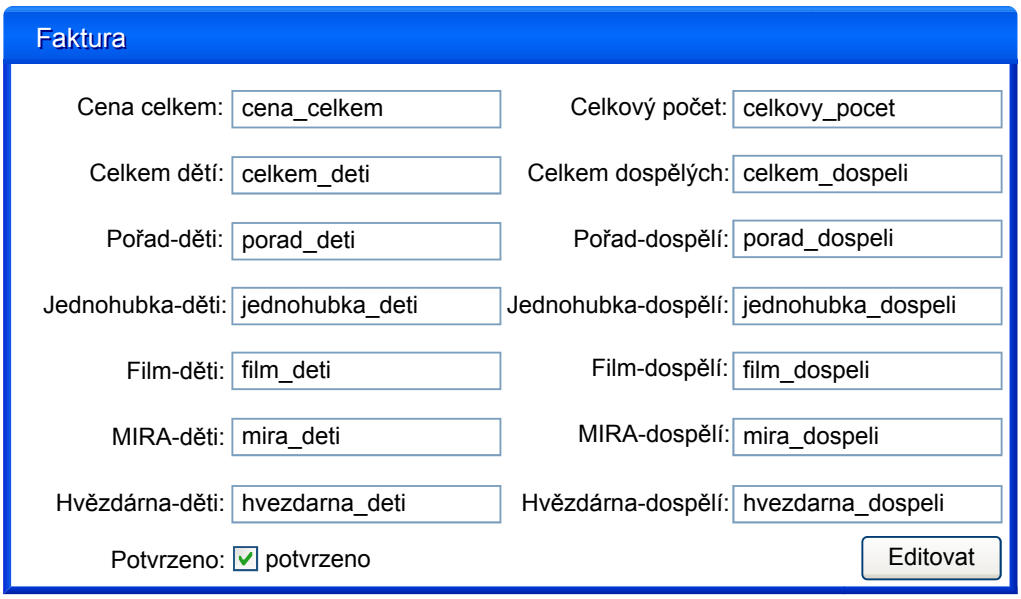
2.9 Výpis faktury … funkce, která poskytuje přehledný výpis vybrané faktury související s konkrétní objednávkou.



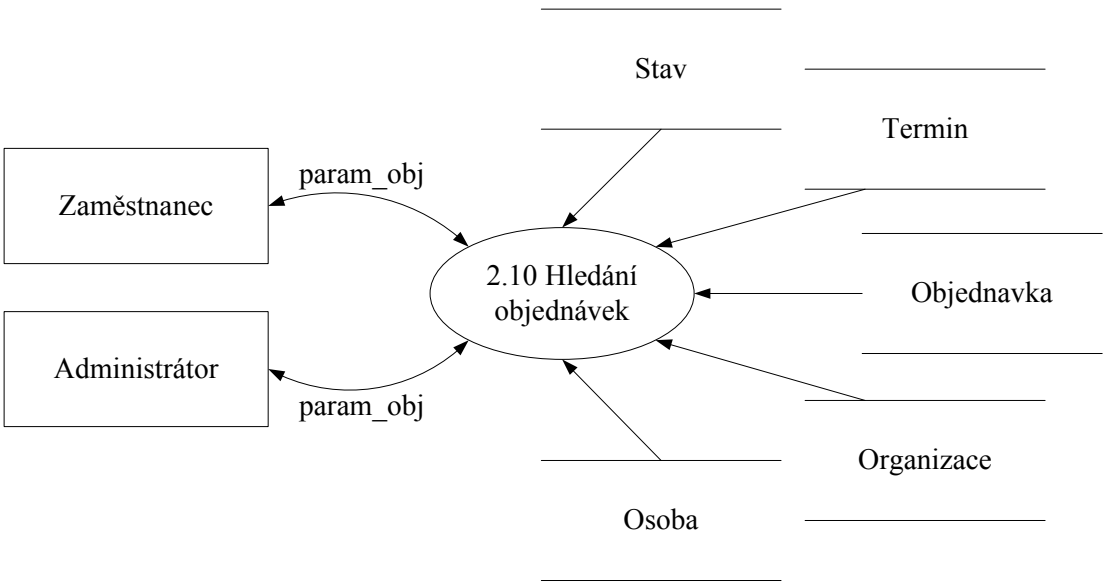
Obrázek 3.15: DFD (2.9 Výpis faktury)

Algoritmus:

1. Zobraz formulář Objednávka (viz. 2.5 Výpis objednávky) s vybranou objednávkou, kde chce uživatel zobrazit fakturu
2. Ulož id_faktura vybrané objednávky do p.id_faktura
3. Uživatel klikne na tlačítko pro zobrazení faktury
4. Načti z tabulky Faktura záznam, kde Faktura.id_faktura = p.id_faktura a ulož atributy do p.potvrzeno, p.cena_celkem, p.celkovy_pocet, p.celkem_deti, p.celkem_dospeli, p.porad_deti, p.porad_dospeli, p.jednohubka_deti, p.jednohubka_dospeli, p.film_deti, p.film_dospeli, p.mira_deti, p.mira_dospeli, p.hvezdarna_deti, p.hvezdarna_dospeli
5. Vypiš záznam ve formuláři Faktura doplněný o odkaz pro editaci faktury



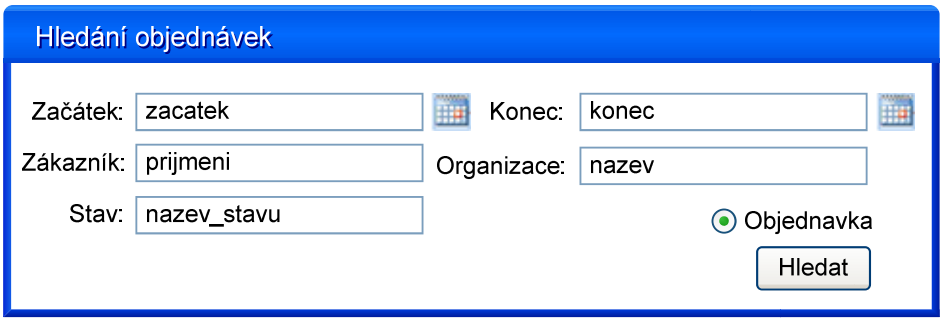
2.10 Hledání objednávek … funkce, která poskytuje možnost vyhledat objednávky v databázi systému podle předem stanovených kritérií.



Obrázek 3.16: DFD (2.10 Hledání objednávek)

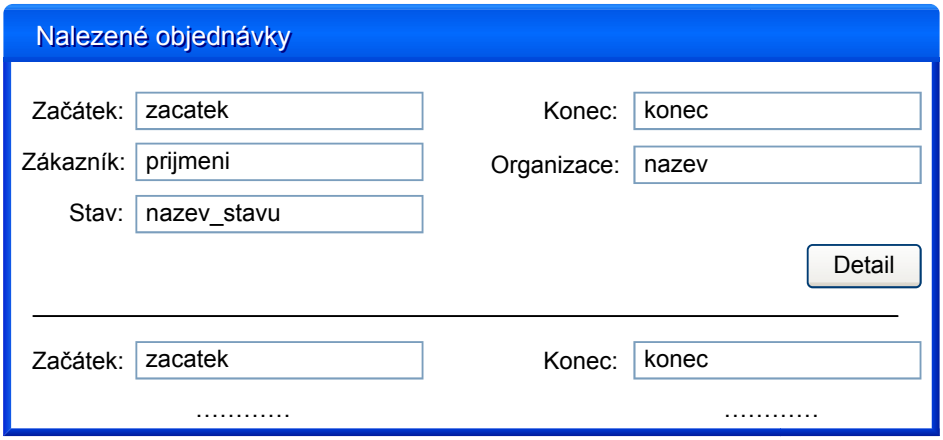
Algoritmus:

1. Zobraz formulář Hledání objednávek

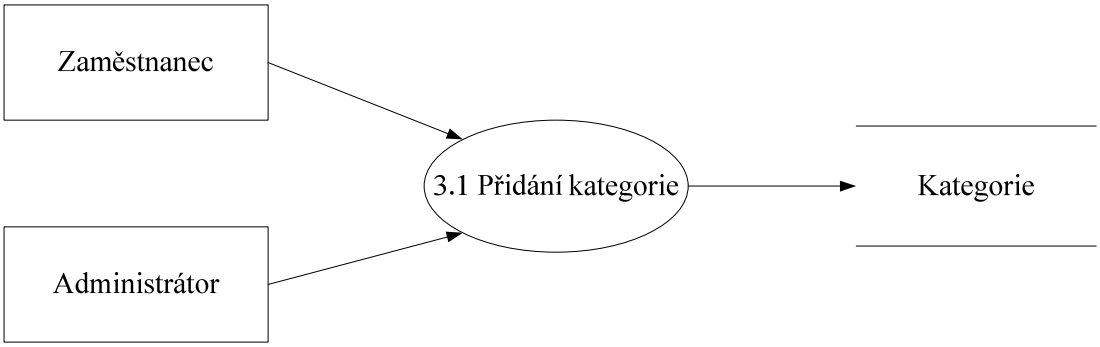


2. Uživatel modifikuje vstupní parametry ve formuláři Hledání objednávek
3. Ulož hodnoty z formuláře Hledání objednávek do p.zacatek, p.konec, p.prijmeni, p.nazev, p.nazev_stavu
4. Uživatel klikne na tlačítko pro vyhledání objednávek podle zadaných parametrů
5. Načti z tabulky Objednavka všechny záznamy, kde Termin.zacatek <= p.zacatek AND Termin.konec <= p.konec AND Termin.id_termin = Objednavka.id_termin AND Osoba.prijmeni LIKE p.prijmeni AND Osoba.id_osoba = Objednavka.id_osoba AND Organizace.nazev LIKE p.nazev AND Organizace.id_organizace = Objednavka.id_organizace AND Stav.nazev_stavu LIKE p.nazev_stavu AND Stav.id_stav = Objednavka.id_stav a ulož hodnoty do p.zacatek, p.konec, p.prijmeni, p.nazev, p.nazev_stavu

6. Vypiš všechny nalezené záznamy do formuláře Nalezené objednávky a zobraz tlačítko pro přechod na detail objednávky



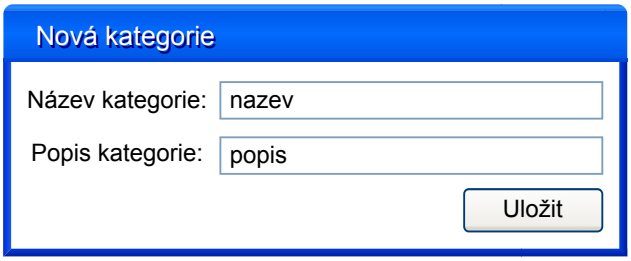
3.1 Přidání kategorie … funkce, která přidává do číselníku novou kategorii posluchačů.



Obrázek 3.17: DFD (3.1 Přidání kategorie)

Algoritmus:

1. Zobraz formulář Nová kategorie



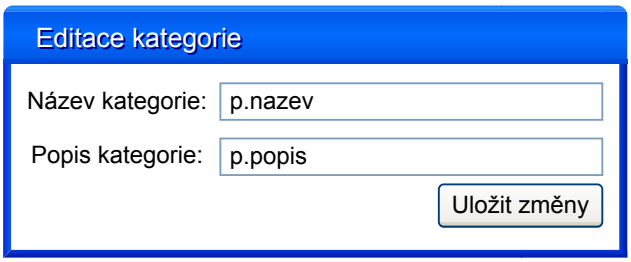
2. Uživatel vyplní formulář Nová kategorie
3. Ulož hodnoty z formuláře Nová kategorie do p.nazev, p.popis
4. Zkontroluj integritní omezení. Jestliže je chyba, vypiš chybovou hlášku a jdi na bod 1
5. Zapiš p.nazev, p.popis do tabulky Kategorie jako nový záznam
6. Jestliže se zápis neprovedl, vypiš chybovou hlášku a jdi na bod 1

3.2 Editace kategorie … funkce, která dovoluje uživateli editovat záznam o kategorii v databázi IS.

Obrázek 3.18: DFD (3.2 Editace kategorie)

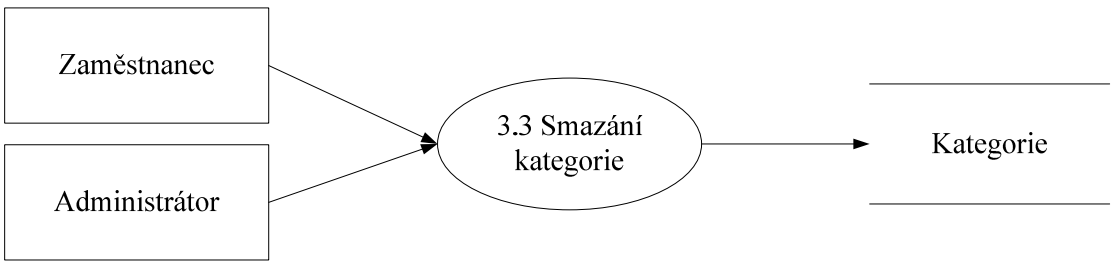
Algoritmus:

1. Zobraz formulář Seznam kategorií (viz. 3.4 Výpis kategorií)
2. Uživatel vybere kategorii, kterou chce editovat
3. Ulož id_kategorie vybrané kategorie do p.id_kategorie
4. Vyber z tabulky Kategorie záznam, kde Kategorie.id_kategorie = p.id_kategorie a ulož hodnoty do p.nazev, p.popis
5. Zobraz formulář Editace kategorie



6. Uživatel modifikuje hodnoty ve formuláři Editace kategorie
7. Ulož hodnoty z formuláře Editace kategorie do p.nazev, p.popis
8. Zkontroluj IO. Jestliže je chyba, vypiš chybovou hlášku a jdi na bod 5
9. Proveď zápis údajů p.nazev, p.popis do tabulky Kategorie u záznamu, kde Kategorie.id_kategorie = p.id_kategorie
10. Jestliže se zápis neprovedl, vypiš chybovou hlášku a jdi na bod 1

3.3 Smazání kategorie … funkce, která odstraní kategorii ze systému.



Obrázek 3.19: DFD (3.3 Smazání kategorie)

Algoritmus:

1. Zobraz formulář Seznam kategorií (viz. 3.4 Výpis kategorií)
2. Uživatel vybere kategorii, kterou chce smazat a klikne na tlačítko pro odstranění záznamu

3. Ulož id_kategorie vybrané kategorie do p.id_kategorie
4. Proveď odstranění záznamu z tabulky Kategorie, kde Kategorie.id_kategorie = p.id_kategorie
5. Jestliže se zápis neprovedl, vypiš chybovou hlášku a jdi na bod 1

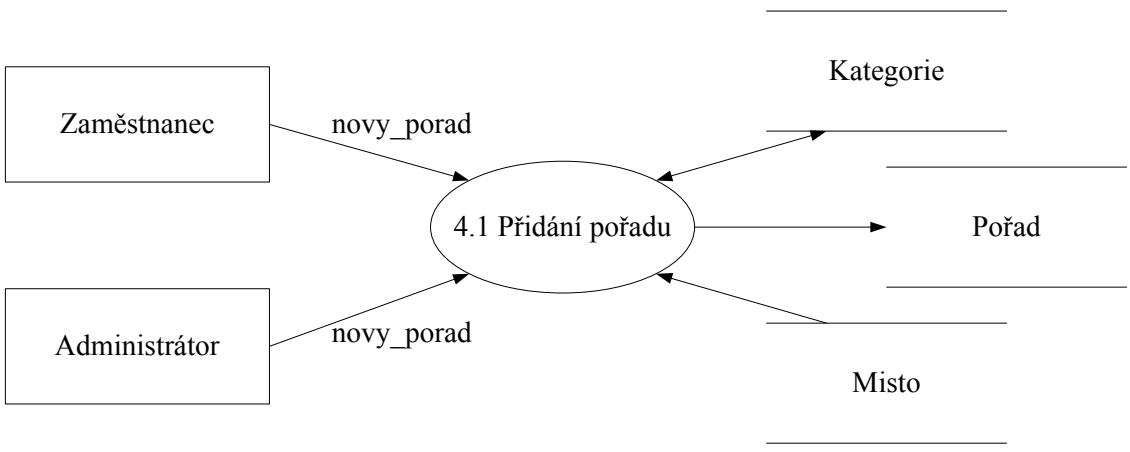
3.4 Výpis kategorií … funkce, která vypíše kategorie z databáze do formuláře Seznam kategorií.

Obrázek 3.20: DFD (3.4 Výpis kategorií)

Algoritmus:

1. Načti z tabulky Kategorie všechny záznamy a setřiď je podle atributu nazev
2. Vypiš záznamy ve formuláři Seznam kategorií podle daného setřídění včetně odkazů pro editaci a smazání kategorie

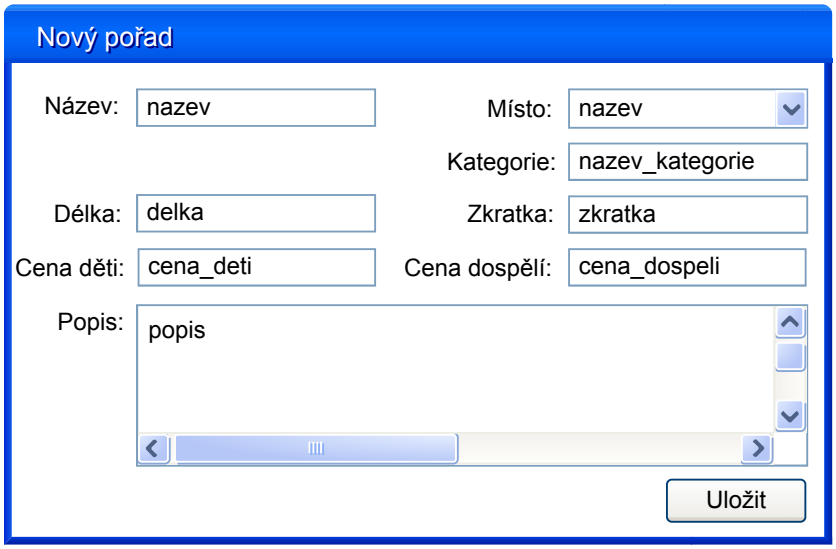
4.1 Přidání pořadu … funkce, která přidává do databáze IS záznam o novém pořadu.



Obrázek 3.21: DFD (4.1 Přidání pořadu)

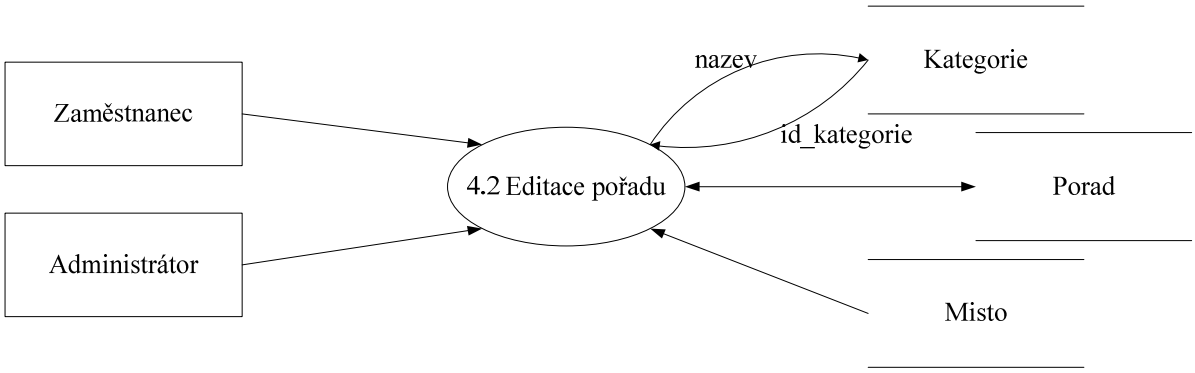
Algoritmus:

1. Zobraz formulář Nový pořad



2. Uživatel vyplní formulář Nový pořad
3. Ulož hodnoty z formuláře Nový pořad do p.nazev, p.delka, p.cena_deti, p.cena_dospeli, p.zkratka, p.popis, p.nazev_kategorie
4. Zkontroluj integritní omezení. Jestliže je chyba, vypiš chybovou hlášku a jdi na bod 1
5. Jestliže chce uživatel změnit ID místa pro nový pořad
   PAK
   5.1. Zobraz seznam míst z tabulky Misto
   5.2. Uživatel vybere místo
   5.3. Ulož id_misto vybraného místa do p.id_misto
6. Jestliže existuje záznam v tabulce Kategorie, kde Kategorie.nazev = p.nazev_kategorie
   PAK
   6.1. Ulož Kategorie.id_kategorie do p.id_kategorie
   JINAK
   6.2. Zapiš p.nazev_kategorie do tabulky Kategorie jako nový záznam a vrať jeho id_kategorie do p.id_kategorie
7. Ulož do p.odstraneno hodnotu FALSE
8. Zapiš p.id_misto, p.id_kategorie, p.nazev, p.delka, p.cena_deti, p.cena_dospeli, p.popis, p.zkratka, p.odstraneno do tabulky Porad jako nový záznam
9. Jestliže se zápis neprovedl, vypiš chybovou hlášku a jdi na bod 1

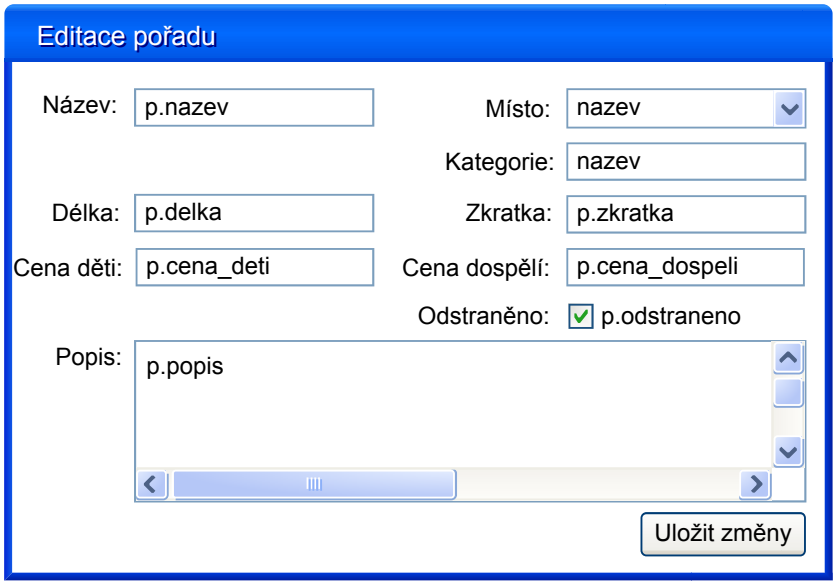
4.2 Editace pořadu … funkce, která dovoluje uživateli editovat záznam o vybraném pořadu v databázi IS. Tato funkce je dostupná pouze pro uživatele typu Administrátor a Zaměstnanec.



Obrázek 3.22: DFD (4.2 Editace pořadu)

Algoritmus:

1. Zobraz formulář Pořad (viz. 4.5 Výpis pořadu) s vybraným pořadem, který chce uživatel editovat
2. Ulož id_porad vybraného pořadu do p.id_porad
3. Uživatel klikne na tlačítko pro editaci pořadu
4. Načti z tabulky Porad záznam, kde Porad.id_porad = p.id_porad a ulož hodnoty do p.id_misto, p.id_kategori, p.delka, p.nazev, p.cena_deti, p.cena_dospeli, p.zkratka, p.popis, p.odstraneno
5. Doplň pořad o atribut nazev z tabulky Kategorie, kde Kategorie.id_kategorie = p.id_kategorie
6. Doplň pořad o atribut nazev z tabulky Misto, kde Misto.id_misto = p.id_misto
7. Vypiš záznam doplněný o výše uvedené atributy ve formuláři Editace pořadu

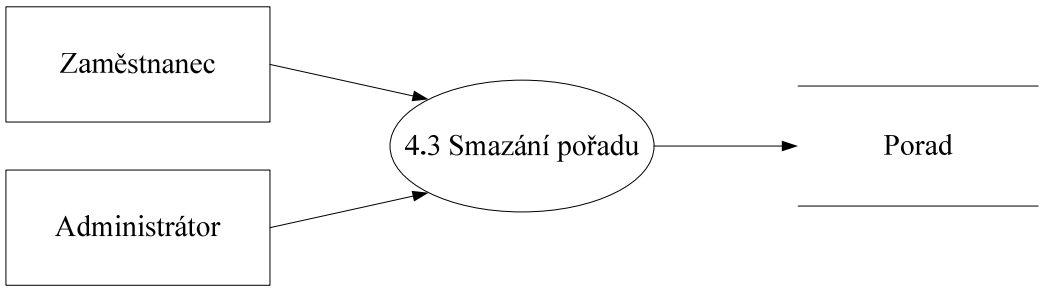


8. Uživatel modifikuje hodnoty ve formuláři Editace pořadu
9. Ulož hodnoty z formuláře Editace pořadu do p.nazev, p.delka, p.cena_deti, p.cena_dospeli, p.zkratka, p.popis, p.nazev_kategorie, p.odstraneno
10. Zkontroluj IO. Jestliže je chyba, vypiš chybovou hlášku a jdi na bod 7
11. Jestliže chce uživatel změnit ID místa pro editovaný pořad

PAK
11.1. Zobraz seznam míst z tabulky Misto
11.2. Uživatel vybere místo
11.3. Ulož id_misto vybraného místa do p.id_misto

12. Jestliže existuje záznam v tabulce Kategorie, kde Kategorie.nazev = p.nazev_kategorie
PAK
12.1. Ulož Kategorie.id_kategorie do p.id_kategorie
JINAK
12.2. Zapiš p.nazev_kategorie do tabulky Kategorie jako nový záznam a vrať jeho id_kategorie do p.id_kategorie
13. Proveď zápis údajů p.id_misto, p.id_kategorie, p.nazev, p.delka, p.cena_deti, p.cena_dospeli, p.popis, p.zkratka, p.odstraneno do tabulky Porad u záznamu, kde Porad.id_porad = p.id_porad
14. Jestliže se zápis neprovedl, vypiš chybovou hlášku a jdi na bod 1

4.3 Smazání pořadu … funkce, která dovoluje uživateli smazat záznam o pořadu z databáze IS.

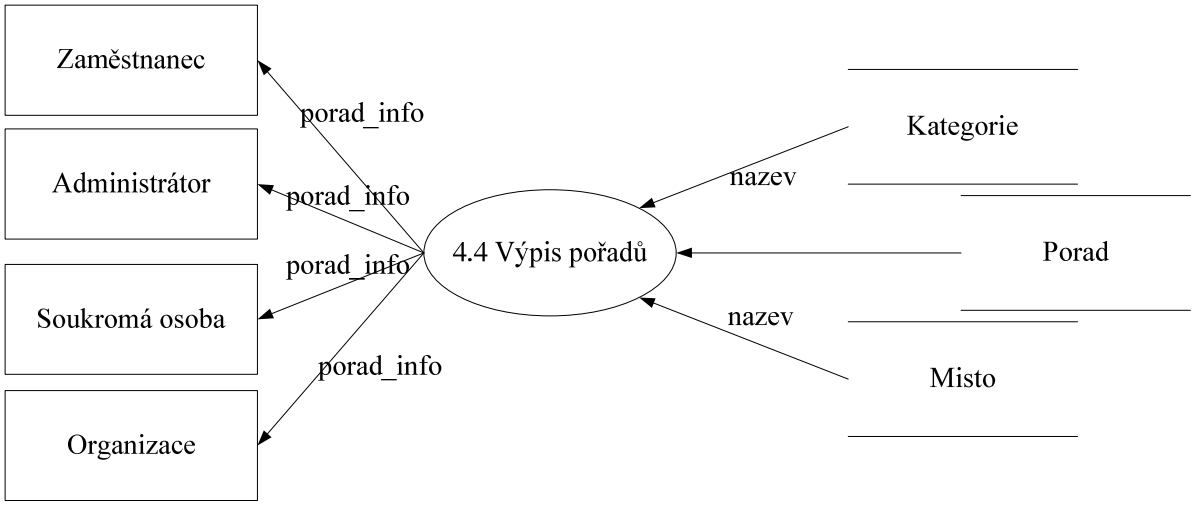


Obrázek 3.23: DFD (4.3 Smazání pořadu)

Algoritmus:

1. Zobraz formulář Pořad (viz. 4.5 Výpis pořadu) s vybraným pořadem, který chce uživatel smazat
2. Ulož id_porad vybraného pořadu do p.id_porad
3. Uživatel klikne na tlačítko pro smazání pořadu
4. Proveď odstranění záznamu z tabulky Porad, kde Porad.id_porad = p.id_porad
5. Jestliže se zápis neprovedl, vypiš chybovou hlášku a jdi na bod 1

4.4 Výpis pořadů … funkce, která poskytuje přehledný výpis všech pořadů z databáze IS.

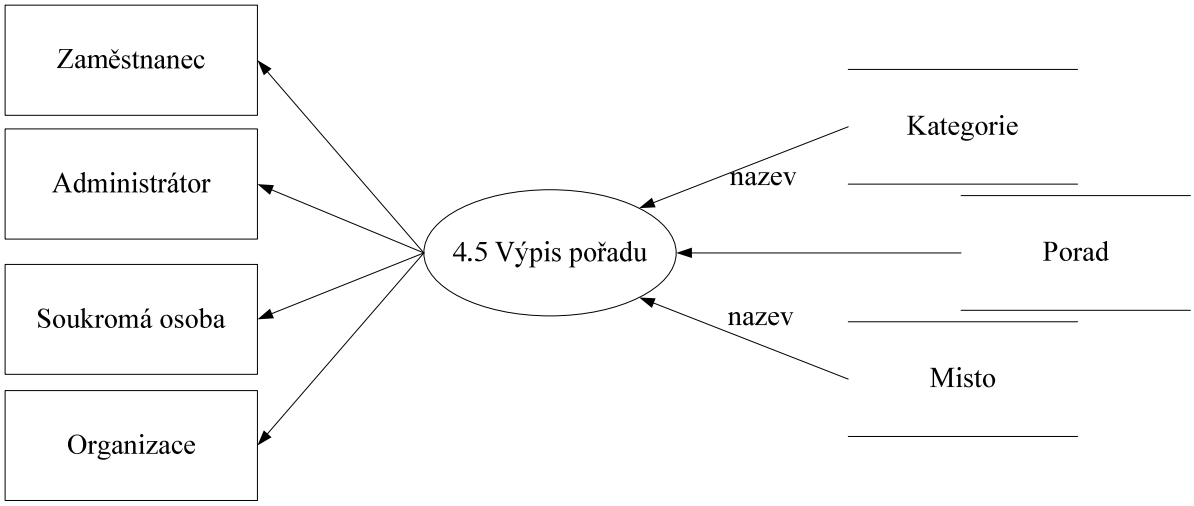


Obrázek 3.24: DFD (4.4 Výpis pořadů)

Algoritmus:

1. Načti z tabulky Porad všechny záznamy a setřiď podle atributu Porad.nazev
2. Doplň každý pořad o atribut nazev z tabulky Kategorie, kde Kategorie.id_kategorie = Porad.id_kategorie
3. Doplň každý pořad o atribut nazev z tabulky Misto, kde Misto.id_misto = Porad.id_misto
4. Vypiš záznamy doplněné o výše uvedené atributy ve formuláři Přehled pořadů podle daného setřídění včetně odkazů pro detail pořadu

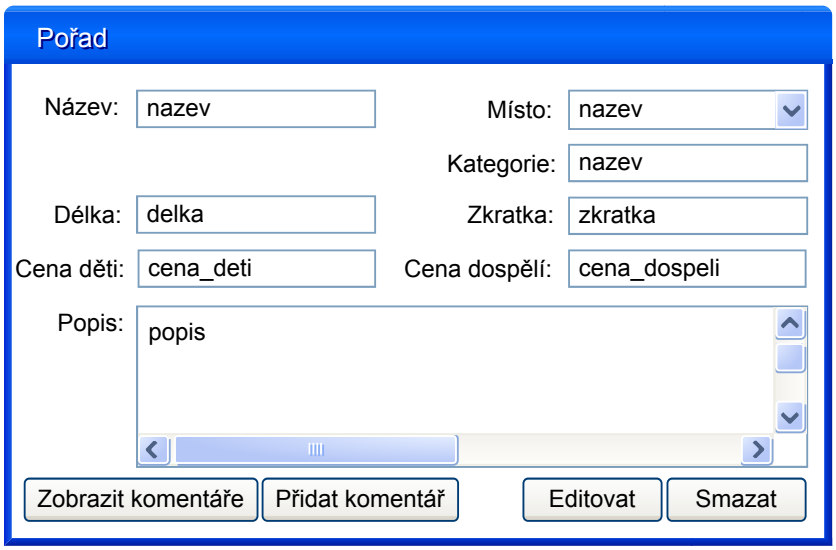
4.5 Výpis pořadu … funkce, která poskytuje výpis detailních informací o pořadu z databáze IS.



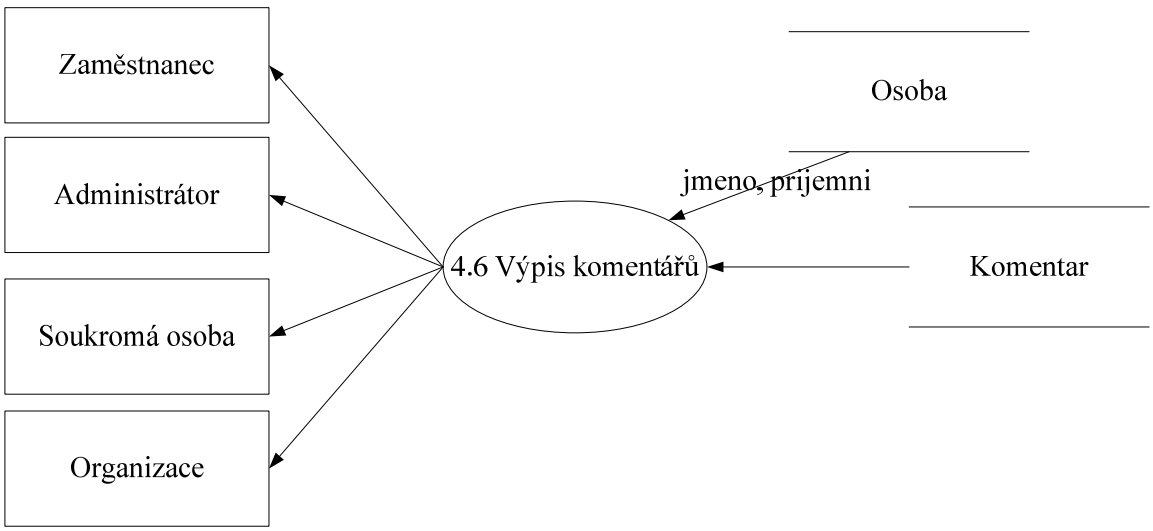
Obrázek 3.25: DFD (4.5 Výpis pořadu)

Algoritmus:

1. Zobraz formulář Přehled pořadů (viz. 4.4 Výpis pořadů)
2. Uživatel vybere pořad, který chce detailně vypsat, kliknutím na tlačítko pro detail
3. Ulož id_porad vybraného pořadu do p.id_porad
4. Načti z tabulky Porad záznam, kde Porad.id_porad = p.id_porad a ulož hodnoty do p.id_misto, p.id_kategorie, p.delka, p.nazev, p.cena_deti, p.cena_dospeli, p.zkratka, p.popis, p.odstraneno
5. Doplň pořad o atribut nazev z tabulky Kategorie, kde Kategorie.id_kategorie = p.id_kategorie
6. Doplň pořad o atribut nazev z tabulky Misto, kde Misto.id_misto = p.id_misto
7. Vypiš záznam doplněný o výše uvedené atributy ve formuláři Pořad včetně odkazů pro další funkce spojené s vybraným pořadem



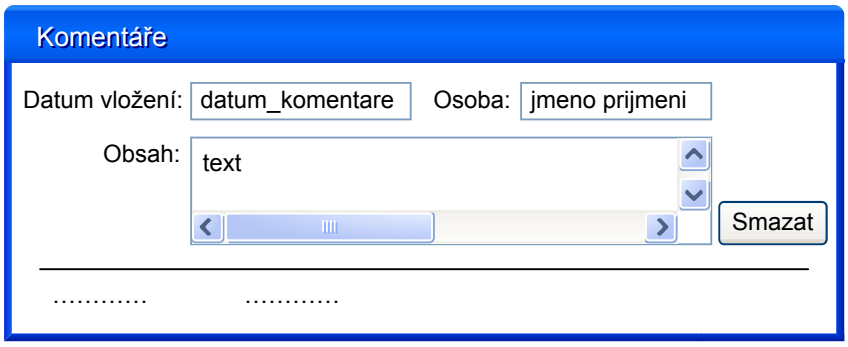
4.6 Výpis komentářů … funkce, která poskytuje uživateli možnost zobrazit všechny dostupné komentáře dalších uživatelů k vybranému pořadu.



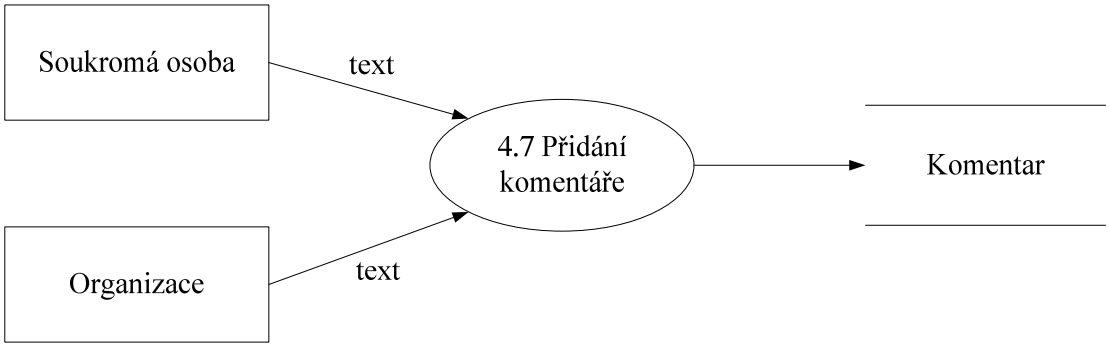
Obrázek 3.26: DFD (4.6 Výpis komentářů)

Algoritmus:

1. Zobraz formulář Pořad (viz. 4.5 Výpis pořadu) s vybraným pořadem, pro který chce uživatel zobrazit komentáře dalších uživatelů
2. Ulož id_porad vybraného pořadu do p.id_porad
3. Uživatel klikne na tlačítko pro zobrazení komentářů
4. Načti z tabulky Komentar všechny záznamy, kde Komentar.id_porad = p.id_porad a setřiď podle atributu datum_komentare
5. Doplň všechny záznamy o atributy jmeno, prijmeni z tabulky Osoba, kde Osoba.id_osoba = Komentar.id_osoba
6. Vypiš všechny záznamy ve formuláři Komentáře podle daného setřídění včetně odkazu pro smazání komentáře



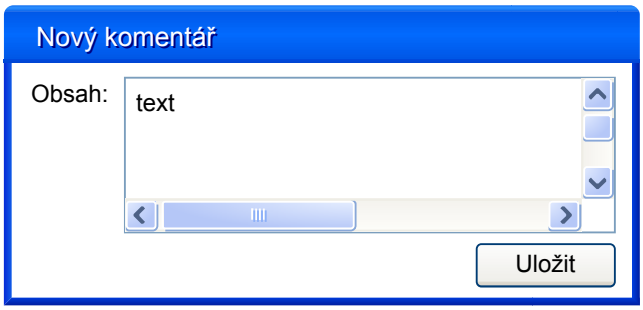
4.7 Přidání komentáře … funkce, která přidává do databáze IS nový komentář k vybranému pořadu.



Obrázek 3.27: DFD (4.7 Přidání komentáře)

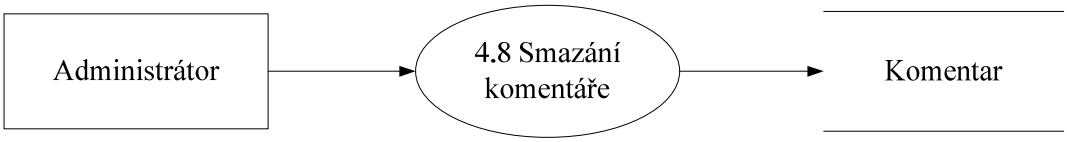
Algoritmus:

1. Zobraz formulář Pořad (viz. 4.5 Výpis pořadu) s vybraným pořadem, který chce uživatel komentovat
2. Ulož id_porad vybraného pořadu do p.id_porad
3. Uživatel klikne na tlačítko pro přidání nového komentáře
4. Zobraz formulář Nový komentář



5. Uživatel vyplní formulář Nový komentář
6. Ulož obsah komentáře z formuláře Nový komentář do p.text
7. Zkontroluj integritní omezení. Jestliže je chyba, vypiš chybovou hlášku a jdi na bod 1
8. Ulož do p.id_osoba hodnotu ID právě přihlášeného uživatele
9. Ulož do p.datum_komentare aktuální systémové datum
10. Zapiš p.id_osoba, p.id_porad, p.datum_komentare, p.text jako nový záznam do tabulky Komentar
11. Jestliže se zápis neprovedl, vypiš chybovou hlášku a jdi na bod 1

4.8 Smazání komentáře … funkce, která dovoluje uživateli v roli Administrátor smazat vybraný komentář z databáze IS. Pokud je komentář vložený uživatelem nějakým způsobem nevhodný apod., potom může být snadno odstraněn.

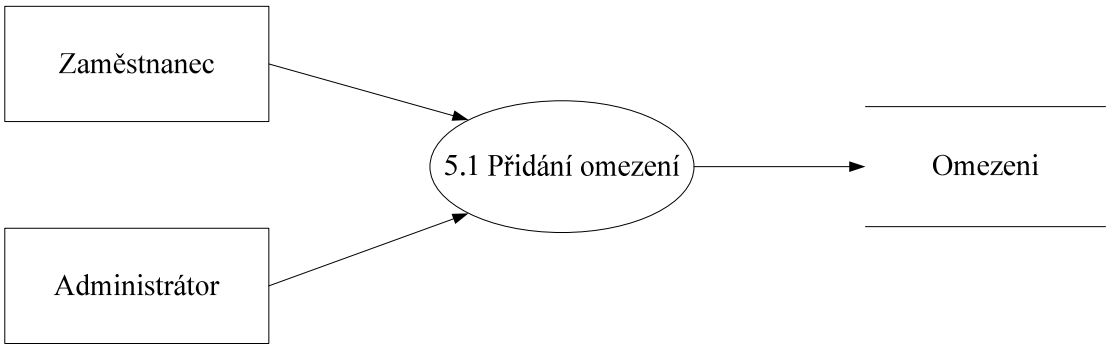


Obrázek 3.28: DFD (4.8 Smazání komentáře)

Algoritmus:

1. Zobraz formulář Komentáře (viz. 4.6 Výpis komentářů)
2. Uživatel vybere komentář, který chce smazat a klikne na tlačítko pro odstranění záznamu
3. Ulož id_komentar vybraného komentáře do p.id_komentar
4. Proveď odstranění záznamu z tabulky Komentar, kde Komentar.id_komentar = p.id_komentar
5. Jestliže se zápis neprovedl, vypiš chybovou hlášku a jdi na bod 1

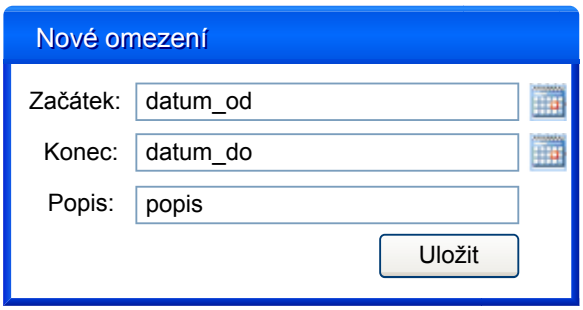
5.1 Přidání omezení … funkce, která přidáva do databáze IS záznam o blokovaném datu, tzv. omezení. Tento údaj je potom použit při kontrolování IO pro novou objednávku, resp. nový program pro veřejnost. Termín těchto záznamů se nesmí shodovat s datem, který je uložen v tabulce Omezení.



Obrázek 3.29: DFD (5.1 Přidání omezení)

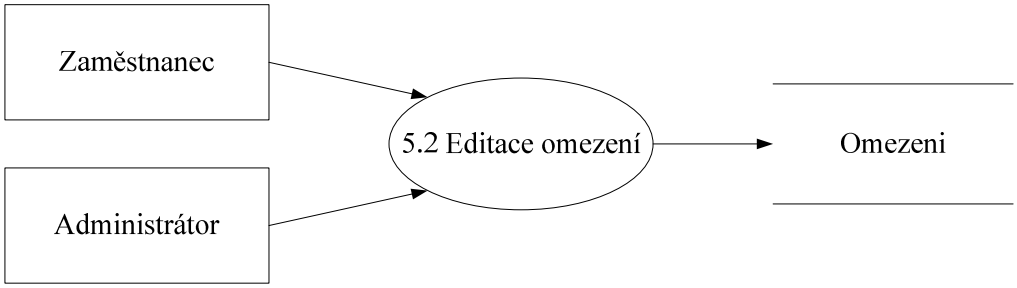
Algoritmus:

1. Zobraz formulář Nové omezení



2. Uživatel vyplní formulář Nové omezení
3. Ulož hodnoty z formuláře Nové omezení do p.datum_od, p.datum_do, p.popis
4. Zkontroluj integritní omezení. Jestliže je chyba, vypiš chybovou hlášku a jdi na bod 1
5. Zapiš p.datum_od, p.datum_do, p.popis jako nový záznam do tabulky Omezeni
6. Jestliže se zápis neprovedl, vypiš chybovou hlášku a jdi na bod 1

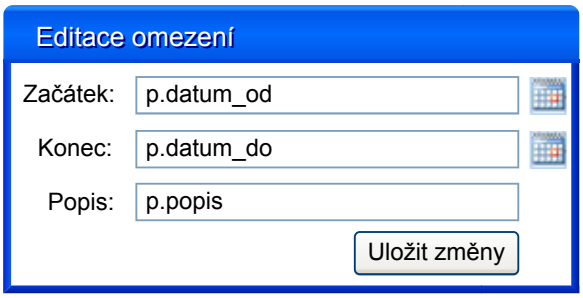
5.2 Editace omezení … funkce, která umožňuje editovat záznam o omezení v databázi IS.



Obrázek 3.30: DFD (5.2 Editace omezení)

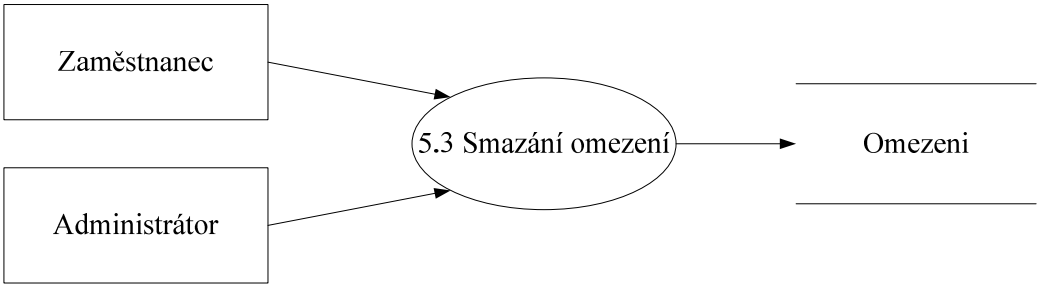
Algoritmus:

1. Zobraz formulář Seznam omezení (viz. 5.4 Výpis omezení)
2. Uživatel vybere omezení, které chce editovat
3. Ulož id_omezeni vybraného omezení do p.id_omezeni
4. Vyber z tabulky Omezeni záznam, kde Omezeni.id_omezeni = p.id_omezeni a ulož hodnoty do p.datum_od, p.datum_do, p.popis
5. Zobraz formulář Editace omezeni



6. Uživatel modifikuje hodnoty ve formuláři Editace omezení
7. Ulož hodnoty z formuláře Editace omezení do p.datum_od, p.datum_do, p.popis
8. Zkontroluj IO. Jestliže je chyba, vypiš chybovou hlášku a jdi na bod 5
9. Proveď zápis údajů p.datum_od, p.datum_do, p.popis do tabulky Omezeni u záznamu, kde Omezeni.id_omezeni = p.id_omezeni
10. Jestliže se zápis neprovedl, vypiš chybovou hlášku a jdi na bod 1

5.3 Smazání omezení … funkce, která odstraní vybraný záznam o omezení z databáze IS.



Obrázek 3.31: DFD (5.3 Smazání omezení)

Algoritmus:

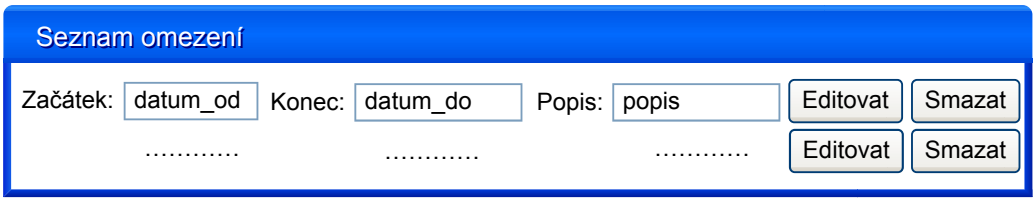
1. Zobraz formulář Seznam omezení (viz. 5.4 Výpis omezení)
2. Uživatel vybere omezení, které chce smazat a klikne na tlačítko pro odstranění záznamu
3. Ulož id_omezeni vybraného omezení do p.id_omezeni
4. Proveď odstranění záznamu z tabulky Omezeni, kde Omezeni.id_omezeni = p.id_omezeni
5. Jestliže se zápis neprovedl, vypiš chybovou hlášku a jdi na bod 1

5.4 Výpis omezení … funkce, která poskytuje uživateli možnost přehledně vypsat všechny záznamy o uložených omezeních v databázi IS spolu s odkazy pro editaci a smazání záznamu.

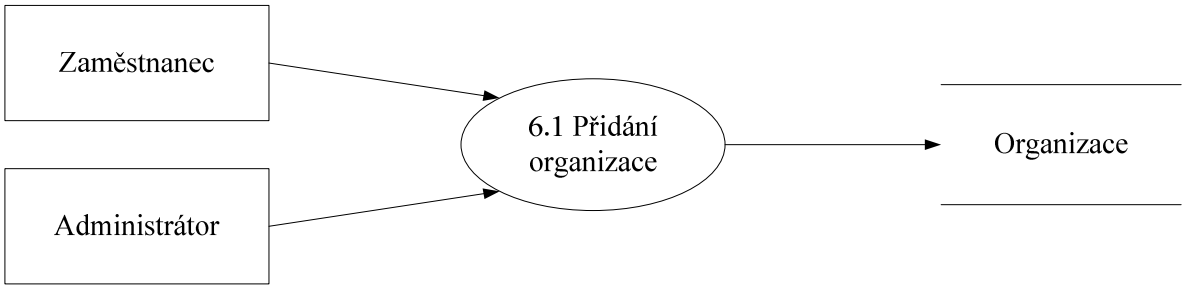
Obrázek 3.32: DFD (5.4 Výpis omezení)

Algoritmus:

1. Načti z tabulky Omezeni všechny záznamy a setřiď je podle atributu datum_od
2. Vypiš záznamy ve formuláři Seznam omezení podle daného setřídění včetně odkazů pro editaci a smazání vybraného omezení



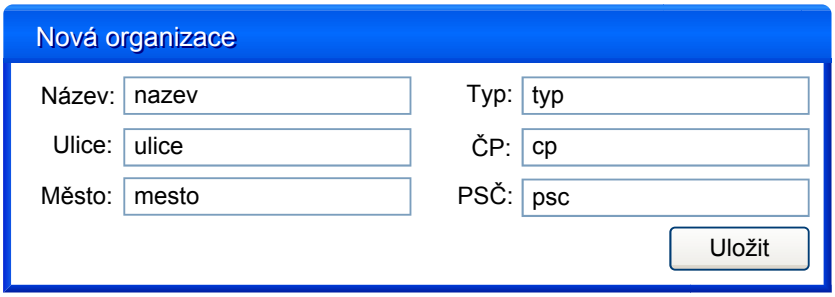
6.1 Přidání organizace … funkce, která přidává do databáze IS nový záznam o organizaci.



Obrázek 3.33: DFD (6.1 Přidání organizace)

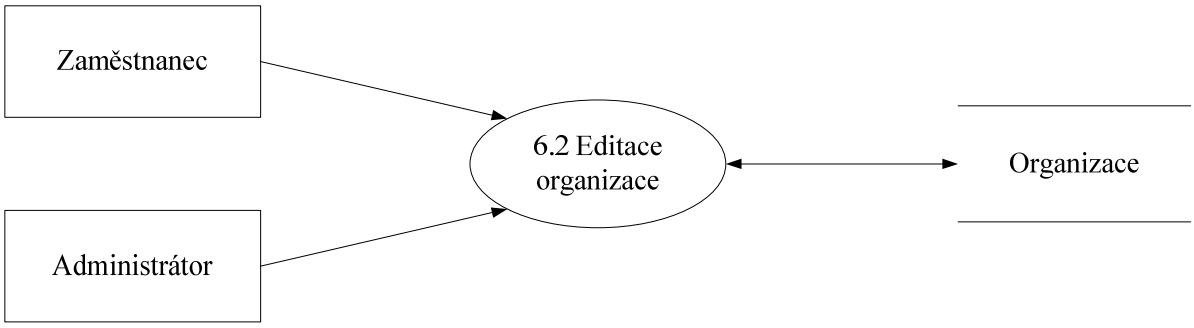
Algoritmus:

1. Zobraz formulář Nová organizace



2. Uživatel vyplní formulář Nová organizace
3. Ulož hodnoty z formuláře Nová organizace do p.nazev, p.typ, p.ulice, p.cp, p.mesto, p.psc
4. Zkontroluj integritní omezení. Jestliže je chyba, vypiš chybovou hlášku a jdi na bod 1
5. Zapiš p.nazev, p.typ, p.ulice, p.cp, p.mesto, p.psc jako nový záznam do tabulky Organizace
6. Jestliže se zápis neprovedl, vypiš chybovou hlášku a jdi na bod 1

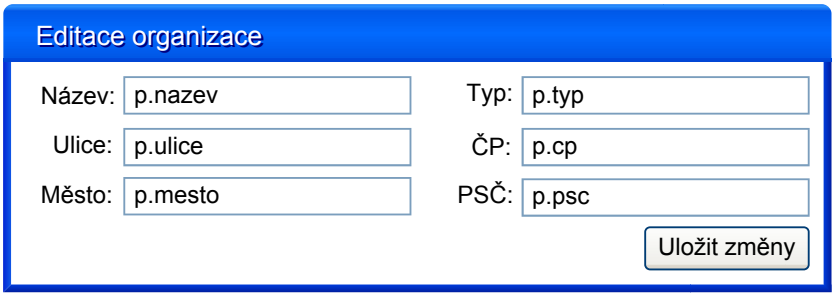
6.2 Editace organizace … funkce, která dovoluje uživateli editovat záznam o vybrané organizaci v databázi IS.



Obrázek 3.34: DFD (6.2 Editace organizace)

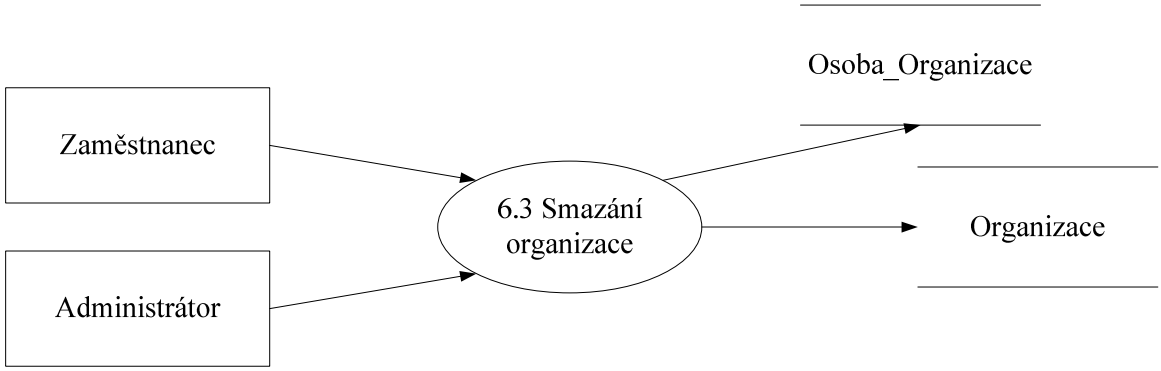
Algoritmus:

1. Zobraz formulář Organizace (viz. 6.5 Výpis organizace) s vybranou organizací, kterou chce uživatel editovat
2. Ulož id_organizace vybrané organizace do p.id_organizace
3. Uživatel klikne na tlačítko pro editaci organizace
4. Načti z tabulky Organizace záznam, kde Organizace.id_organizace = p.id_organizace a ulož hodnoty do p.id_organizace, p.nazev, p.typ, p.ulice, p.cp, p.mesto, p.psc
5. Zobraz formulář Editace organizace



6. Uživatel modifikuje hodnoty ve formuláři Editace organizace
7. Ulož hodnoty z formuláře Editace organizace do p.nazev, p.typ, p.ulice, p.cp, p.mesto, p.psc
8. Zkontroluj IO. Jestliže je chyba, vypiš chybovou hlášku a jdi na bod 5
9. Proveď zápis údajů p.nazev, p.typ, p.ulice, p.cp, p.mesto, p.psc do tabulky Organizace u záznamu, kde Organizace.id_organizace = p.id_organizace
10. Jestliže se zápis neprovedl, vypiš chybovou hlášku a jdi na bod 1

6.3 Smazání organizace … funkce, která odstraní záznam o organizaci z databáze IS. Uživatel může provést také odstranění pouze související organizace z databáze IS. Pro tuto funkci je třeba provést výpis souvisejích organizací (viz. 6.4 Výpis organizací) a potom musí uživatel kliknout na tlačítko pro odstranění související organizac.



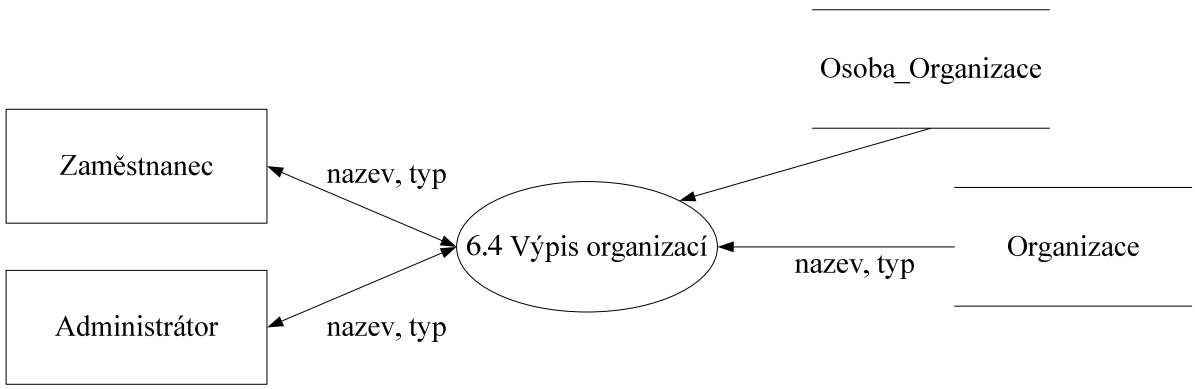
Obrázek 3.35: DFD (6.3 Smazání organizace)

Algoritmus:

1. Zobraz formulář Organizace (viz. 6.5 Výpis organizace) s vybranou organizací, kterou chce uživatel smazat
2. Ulož id_organizace vybrané organizace do p.id_organizace
3. Uživatel klikne na tlačítko pro smazání organizace
4. Proveď odstranění záznamu z tabulky Organizace, kde Organizace.id_organizace = p.id_organizace
5. Jestliže se zápis neprovedl, vypiš chybovou hlášku a jdi na bod 1
6. Jestliže chce uživatel odstranit pouze související organizaci (tj. vazbu na osobu)
   PAK
   6.1. Proveď funkci Výpis organizací (viz. 6.4 Výpis organizací), který vypíše související organizace
   6.2. Uživatel klikne na tlačítko pro odstranění související organizace
   6.3. Ulož id_organizace do p.id_organizace
   6.4. Ulož id_osoba, pro kterou jsou vypsány související organizace, do p.id_osoba
   6.5. Proveď odstranění záznamu z tabulky Osoba_Organizace, kde Osoba_Organizace.id_organizace = p.id_organizace AND Osoba_Organizace.id_osoba = p.id_osoba
   6.6. Jestliže se zápis neprovedl, vypiš chybovou hlášku a jdi na bod 6

6.4 Výpis organizací … funkce, která poskytuje uživateli možnost vypsat organizace z databáze IS podle zadaných filtrovacích kritérií. Po kliknutí na tlačítko pro detail organizace je uživateli zobrazen

formulář s detailními informacemi o vybrané organizaci. Pokud chce uživatel vypsat organizace související s určitou osobou, je třeba zadat ID vybrané osoby (zákazníka) a výpis organizací bude obsahovat pouze organizace, které jsou ve vazbě k této osobě.



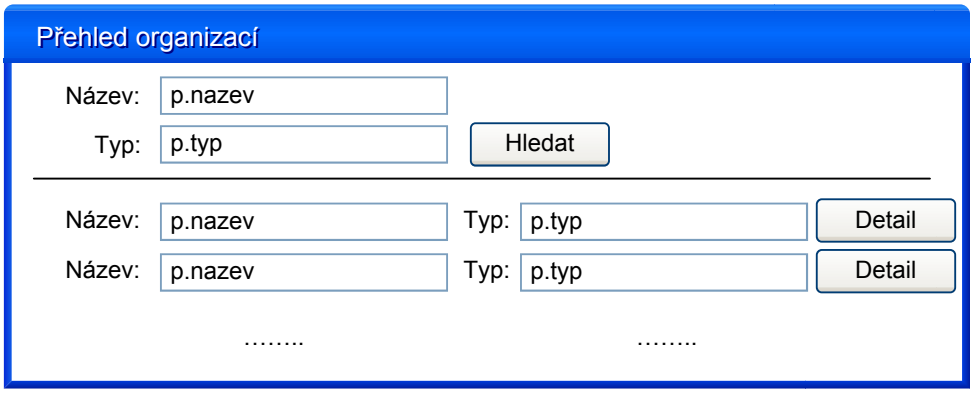
Obrázek 3.36: DFD (6.4 Výpis organizací)

Algoritmus:

1. Zobraz záhlaví formuláře Přehled organizací

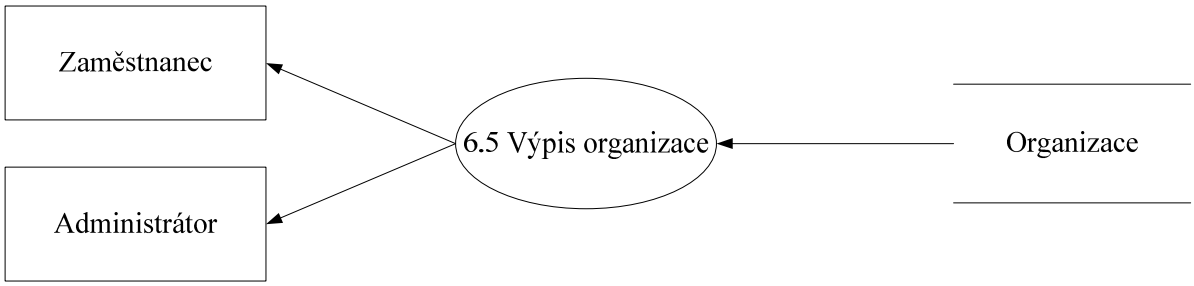
2. Uživatel vyplní hodnoty v záhlaví formuláře Přehled organizací, které budou použity pro výpis hledaných organizací
3. Ulož hodnoty ze záhlaví formuláře Přehled organizací do p.nazev, p.typ
4. Jestliže p.nazev = NULL
   PAK
   4.1. Ulož do p.nazev hodnotu ‘%‘
5. Jestliže p.typ = NULL
   PAK
   5.1. Ulož do p.typ hodnotu ‘%‘
6. Načti z tabulky Organizace všechy záznamy, kde Organizace.nazev LIKE p.nazev AND Organizace.typ LIKE p.typ a ulož do p.nazev, p.typ

7. Vypiš záznamy ve formuláři Přehled organizací



8. Jestliže chce uživatel zobrazit související organizace k vybrané osobě
   PAK
   8.1. Proveď funkci Výpis osoby (6.10 Výpis osoby)
   8.2. Ulož hodnotu id_osoba vybrané osoby do p.id_osoba
   8.3. Uživatel klikne na tlačítko pro zobrazení souvisejících organizací
   8.4. Načti z tabulky Organizace všechny záznamy, kde Organizace.id_organizace = Osoba_Organizace.id_organizace AND Osoba_Organizace.id_osoba = p.id_osoba a ulož do p.nazev, p.typ
   8.5. Vypiš záznamy ve formuláři Přehled organizací a zobraz tlačítko pro odstranění související organizace

6.5 Výpis organizace … funkce, který vypíše z database IS detailní informace o vybrané organizaci.

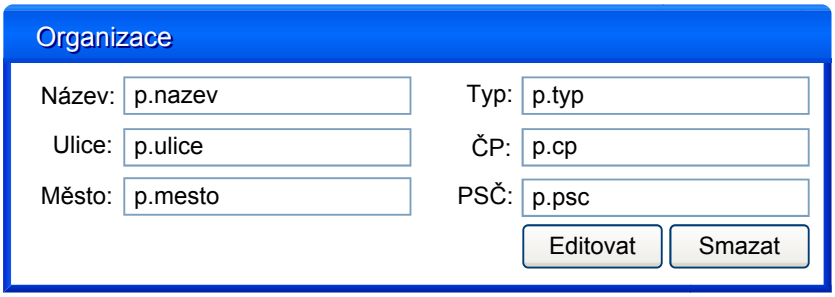


Obrázek 3.37: DFD (6.5 Výpis organizace)

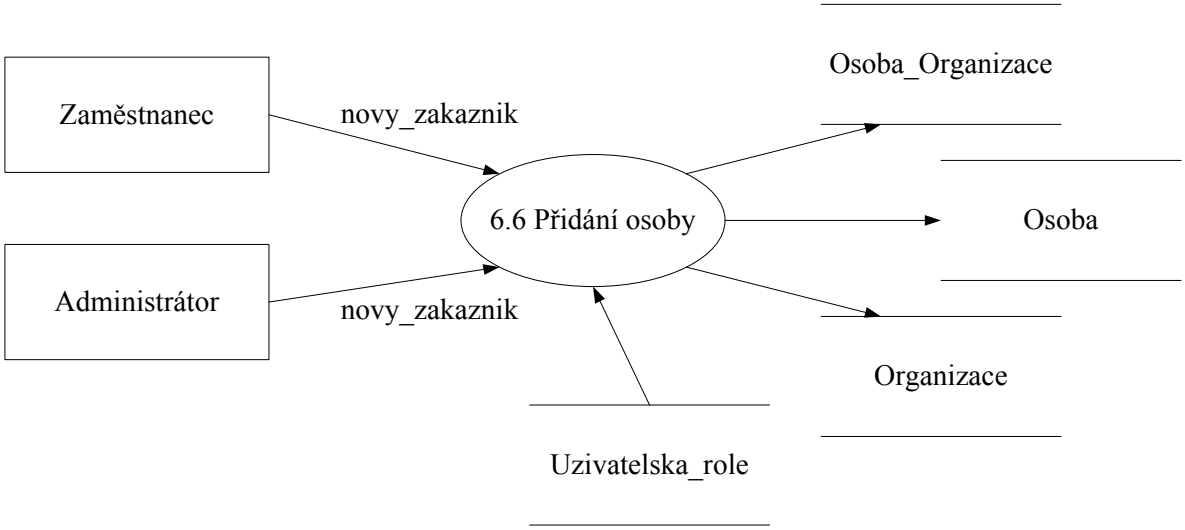
Algoritmus:

1. Zobraz formulář Přehled organizací (viz. 6.4 Výpis organizací)
2. Uživatel vybere organizaci, kterou chce detailně vypsat, kliknutím na tlačítko pro detail
3. Ulož id_organizace vybrané organizace do p.id_organizace
4. Načti z tabulky Organizace záznam, kde Organizace.id_organizace = p.id_organizace a ulož hodnoty do p.id_organizace, p.nazev, p.typ, p.ulice, p.cp, p.mesto, p.psc

5. Vypiš záznam ve formuláři Organizace včetně odkazů pro editaci a smazání organizace



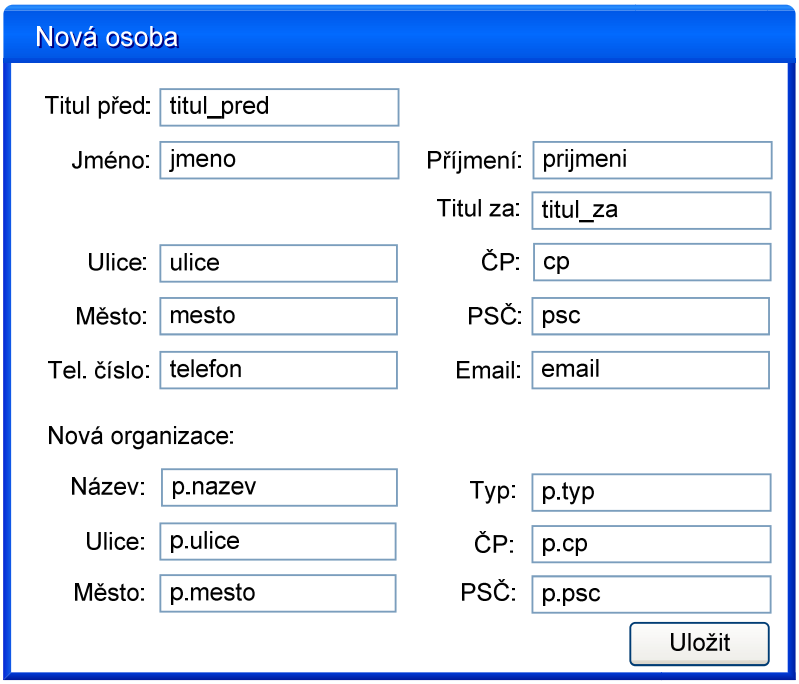
6.6 Přidání osoby … funkce, která přidává do databáze IS nový záznam o osobě (zákazníkovi). Tato funkce navíc dovoluje přidat rovnou i související organizaci, pokud přidávaná osoba vystupuje v systému za nějakou organizaci.



Obrázek 3.38: DFD (6.6 Přidání osoby)

Algoritmus:

1. Zobraz formulář Nová osoba



2. Uživatel vyplní formulář Nová osoba
3. Ulož hodnoty z formuláře Nová osoba do p.titul_pred, p.jmeno, p.prijmeni, p.titul_za, p1.ulice, p1.cp, p1.mesto, p1.psc, p.telefon, p.email, p.nazev, p.typ, p2.ulice, p2.cp, p2.mesto, p2.psc
4. Zkontroluj integritní omezení. Jestliže je chyba, vypiš chyvovou hlášku a jdi na bod 1
5. BEGIN TRANSACTION
6. Jestliže p.nazev, p.typ IS NOT NULL

PAK

6.1. Načti z tabulky Organizace všechny záznamy, kde Organizace.nazev = p.nazev AND Organizace.typ = p.typ a ulož hodnoty do s.id_organizace, s.nazev, s.typ, s.ulice, s.cp, s.mesto, s.psc

6.2. Jestliže neexistuje záznam v tabulce Organizace, kde Organizace.nazev = p.nazev AND Organizace.typ = p.typ

PAK

6.2.1. Zapiš hodnoty p.nazev, p.typ, p2.ulice, p2.cp, p2.mesto, p2.psc do tabulky Organizace jako nový záznam a vrať jeho id_organizace do p.id_organizace

JINAK

6.2.2. Zobraz všechny záznamy z tabulky Organizace, kde kde Organizace.nazev = p.nazev AND Organizace.typ = p.typ

6.2.3. Uživatel vybere organizaci, kterou chce přidat jako související organizaci

6.2.4. Ulož id_organizace vybrané organizace do p.id_organizace

6.3. Ulož do p.id_role hodnotu id_role z tabulky Uzivatelska_role, kde Uzivatelska_role.nazev_role = „Organizace“

JINAK

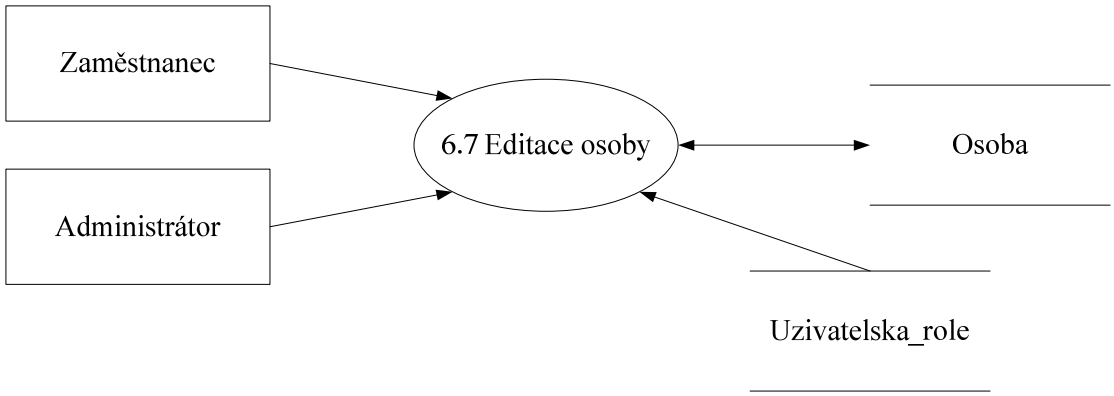
6.4. Ulož do p.id_role hodnotu id_role z tabulky Uzivatelska_role, kde Uzivatelska_role.nazev_role = „Soukromá osoba“

7. Ulož do p.id_osoba nové ID generované z hodnoty p.prijmeni
8. Zapiš hodnoty p.id_osoba, p.id_role, p.titul_pred, p.jmeno, p.prijmeni, p.titul_za, p1.ulice, p1.cp, p1.mesto, p1.psc, p.telefon, p.email do tabulky Osoba jako nový záznam
9. Jestliže p.id_role = Uzivatelska_role.id_role, kde Uzivatelska_role.nazev_role = „Organizace“

   PAK

   9.1. Zapiš hodnoty p.id_organizace, p.id_osoba do tabulky Osoba_Organizace jako nový záznam
10. END TRANSACTION
11. Jestliže se zápis neprovedl, vypiš chybovou hlášku a jdi na bod 1

6.7 Editace osoby … funkce, která dovoluje editovat informace o osobě v databázi IS.

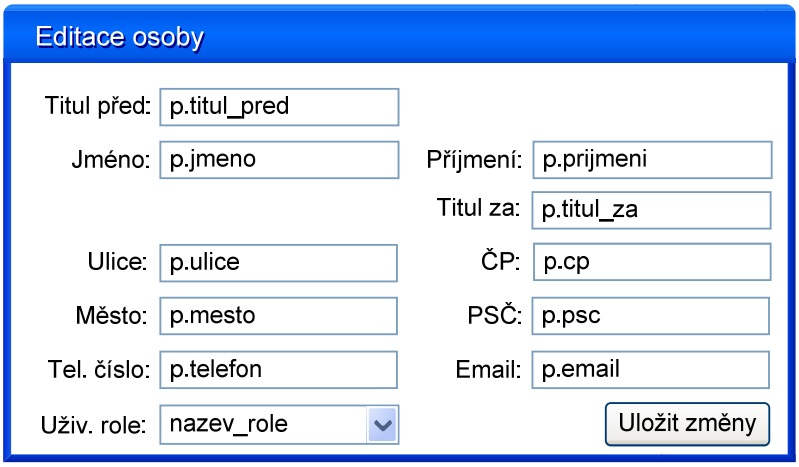


Obrázek 3.39: DFD (6.7 Editace osoby)

Algoritmus:

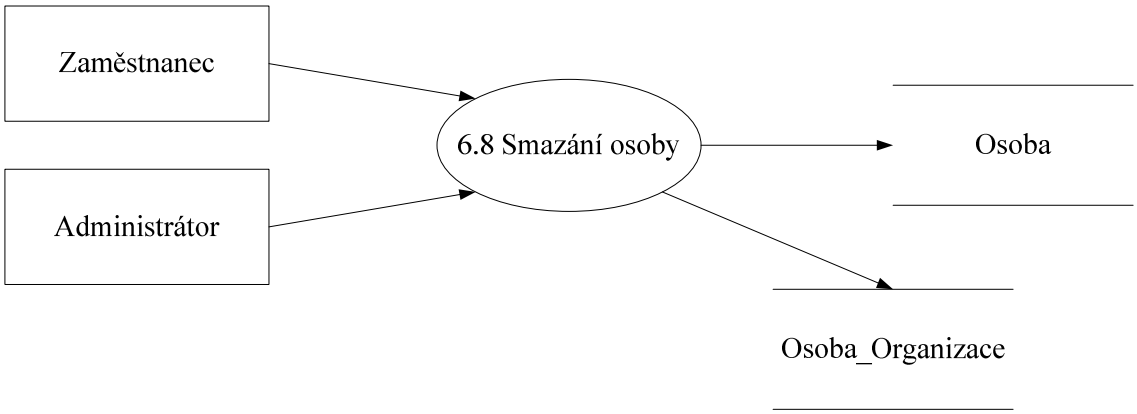
1. Zobraz formulář Osoba (viz. 6.10 Výpis osoby) s vybranou osobou, kterou chce uživatel editovat
2. Ulož id_osoba vybrané osoby do p.id_osoba
3. Uživatel klikne na tlačítko pro editaci osoby
4. Načti z tabulky Osoba záznam, kde Osoba.id_osoba = p.id_osoba a ulož hodnoty do p.id_role, p.titul_pred, p.jmeno, p.prijmeni, p.titul_za, p.ulice, p.cp, p.mesto, p.psc, p.telefon, p.email
5. Doplň osobu o atribut nazev_role z tabulky Uzivatelska_role, kde Uzivatelska_role.id_role = p.id_role

6. Vypiš záznam doplněný o výše uvedený atribut ve formuláři Editace osoby



7. Uživatel modifikuje hodnoty ve formuláři Editace osoby
8. Ulož hodnoty z formuláře Editace osoby do p.titul_pred, p.jmeno, p.prijmeni, p.titul_za, p.ulice, p.cp, p.mesto, p.psc, p.telefon, p.email
9. Zkontroluj IO. Jestliže je chyba, vypiš chybovou hlášku a jdi na bod 6
10. Jestliže chce uživatel změnit ID uživatelské role pro editovanou osobu
    PAK
    10.1. Zobraz seznam uživatelských rolí z tabulky Uzivatelska_role
    10.2. Uživatel vybere uživatelskou roli
    10.3. Ulož id_role vybrané uživatelské role do p.id_role
11. Proveď zápis údajů p.id_role, p.titul_pred, p.jmeno, p.prijmeni, p.titul_za, p.ulice, p.cp, p.mesto, p.psc, p.telefon, p.email do tabulky Osoba u záznamu, kde Osoba.id_osoba = p.id_osoba
12. Jestliže se zápis neprovedl, vypiš chybovou hlášku a jdi na bod 1

6.8 Smazání osoby … funkce, která odstraní záznam o vybrané osobě z databáze IS.

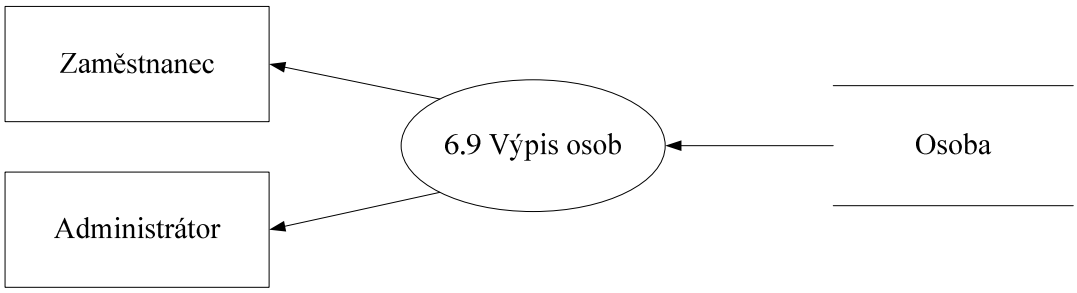


Obrázek 3.40: DFD (6.8 Smazání osoby)

Algoritmus:

1. Zobraz formulář Osoba (viz. 6.10 Výpis osoby) s vybranou osobou, kterou chce uživatel smazat
2. Ulož id_osoba vybrané osoby do p.id_osoba
3. Uživatel klikne na tlačítko pro smazání osoby
4. BEGIN TRANSACTION
5. Proveď odstranění záznamu z tabulky Osoba_Organizace, kde Osoba_Organizace.id_osoba = p.id_osoba
6. Proveď odstranění záznamu z tabulky Osoba, kde Osoba.id_osoba = p.id_osoba
7. END TRANSACTION
8. Jestliže se zápis neprovedl, vypiš chybovou hlášku a jdi na bod 1

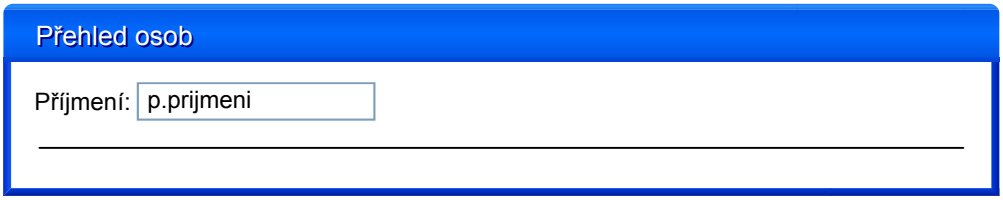
6.9 Výpis osob … funkce, která poskytuje přehledný výpis všech osob z databáze IS podle zadaných vstupních kritérií.



Obrázek 3.41: DFD (6.9 Výpis osob)

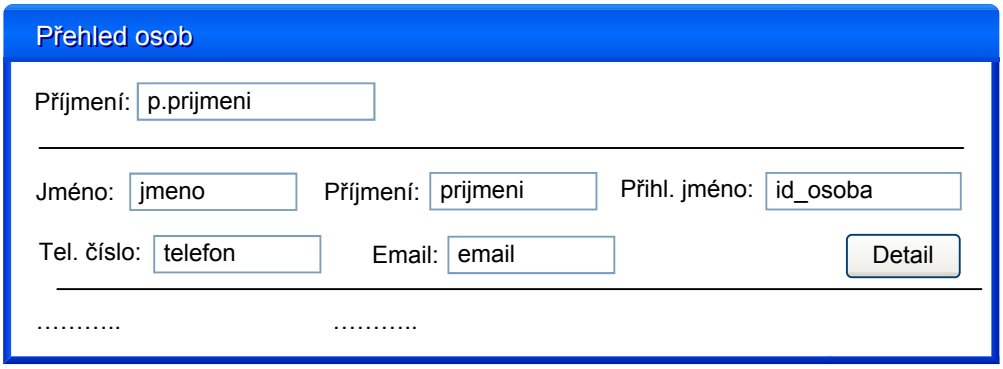
Algoritmus:

1. Zobraz záhlaví formuláře Přehled osob

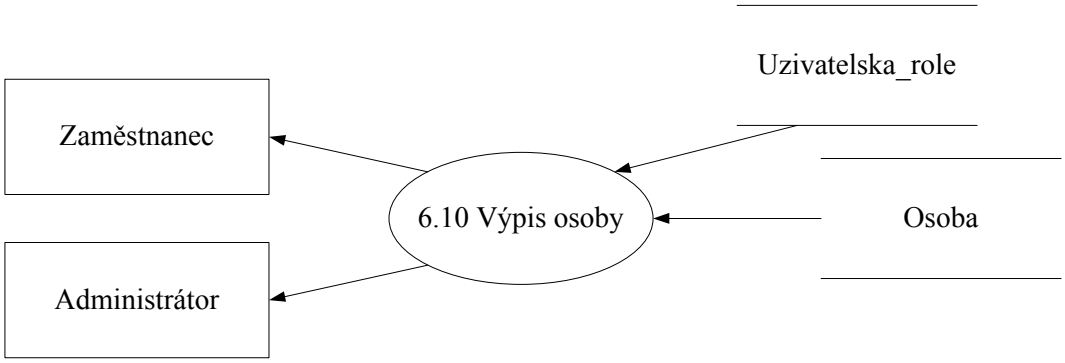


2. Uživatel vyplní hodnotu v záhlaví formuláře Přehled osob
3. Ulož hodnotu ze záhlaví formuláře Přehled osob do p.prijmeni
4. Jestliže p.prijmeni = NULL
   PAK
   4.1. Ulož do p.prijmeni hodnotu ‘%’
5. Načti z tabulky Osoba všechny záznamy, kde Osoba.prijmeni LIKE p.prijmeni a setřiď podle atributu jmeno, prijmeni

6. Vypiš záznamy ve formuláři Přehled osob podle daného setřídění včetně odkazů pro detail osoby



6.10 Výpis osoby … funkce, která poskytuje výpis detailních informací o vybrané osobě z databáze IS.

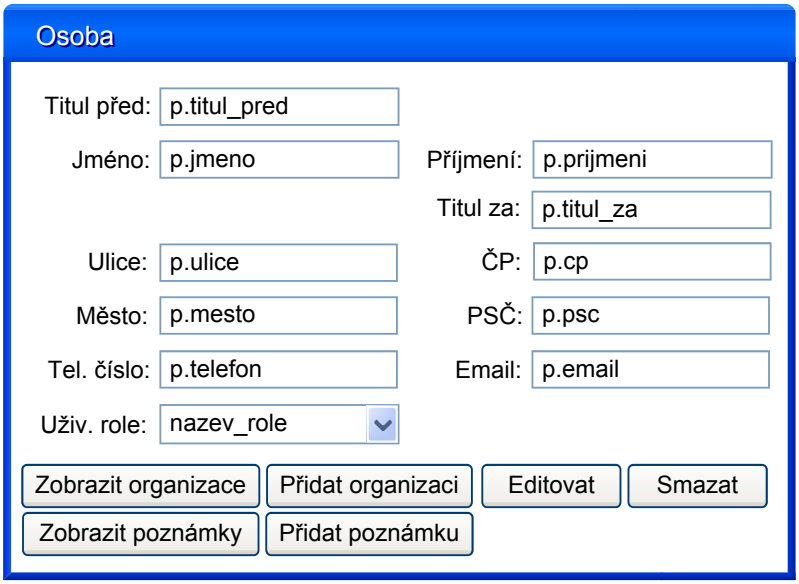


Obrázek 3.42: DFD (6.10 Výpis osob)

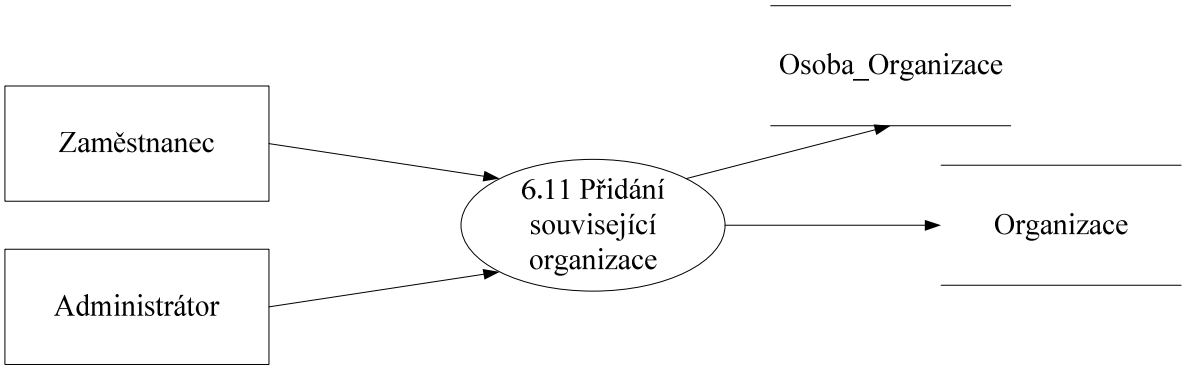
Algoritmus:

1. Zobraz formulář Přehled osob (viz. 6.9 Výpis osob)
2. Uživatel vybere osobu, kterou chce detailně zobrazit na obrazovku, kliknutím na tlačítko pro detail
3. Ulož id_osoba vybrané osoby do p.id_osoba
4. Načti z tabulky Osoba záznam, kde Osoba.id_osoba = p.id_osoba a ulož hodnoty do p.id_role, p.titul_pred, p.jmeno, p.prijmeni, p.titul_za, p.ulice, p.cp, p.mesto, p.psc, p.telefon, p.email
5. Doplň osobu o atribut nazev_role z tabulky Uzivatelska_role, kde Uzivatelska_role.id_role = p.id_role

6. Vypiš záznam doplněný o výše uvedený atribut ve formuláři Osoba včetně odkazů pro další funkce spojené s vybranou osobou



6.11 Přidání související organizace … funkce, která do databáze IS přidá vazební záznam mezi specifikovanou organizací a vybranou osobou.

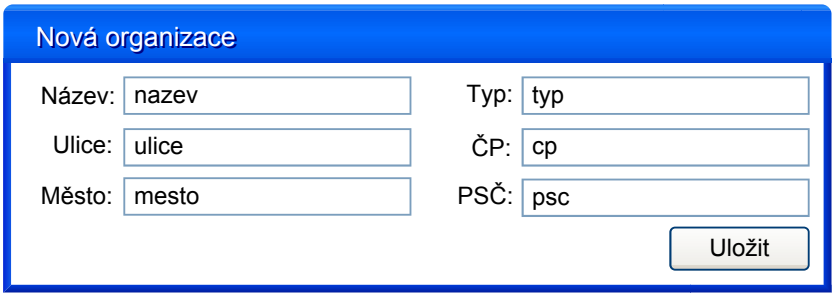


Obrázek 3.43: DFD (6.11 Přidání související organizace)

Algoritmus:

1. Zobraz formulář Osoba (viz. 6.10 Výpis osoby) s vybranou osobo, ke které chce uživatel přidat související organizaci
2. Ulož id_osoba vybrané osoby do p.id_osoba
3. Uživatel klikne na tlačítko pro přidání související organizace

4. Zobraz formulář Nová organizace



5. Uživatel vyplní formulář Nová organizace
6. Ulož hodnoty z formuláře Nová organizace do p.nazev, p.typ, p.ulice, p.cp, p.mesto, p.psc
7. Zkontroluj integritní omezení. Jestliže je chyba, vypiš chybovou hlášku a jdi na bod 4
8. Načti z tabulky Organizace všechny záznamy, kde Organizace.nazev = p.nazev AND Organizace.typ = p.typ a ulož hodnoty do s.id_organizace, s.nazev, s.typ, s.ulice, s.cp, s.mesto, s.psc
9. BEGIN TRANSACTION
10. Jestliže neexistuje záznam v tabulce Organizace, kde Organizace.nazev = p.nazev AND Organizace.typ = p.typ

    PAK

    10.1. Zapiš hodnoty p.nazev, p.typ, p.ulice, p.cp, p.mesto, p.psc do tabulky Organizace jako nový záznam a vrať jeho id_organizace do p.id_organizace

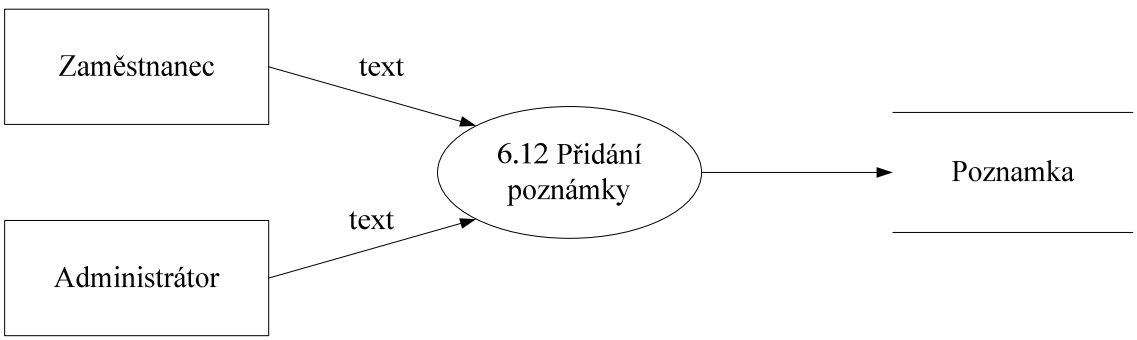
    JINAK

    10.2. Zobraz všechny záznamy z tabulky Organizace, kde kde Organizace.nazev = p.nazev AND Organizace.typ = p.typ

    10.3. Uživatel vybere organizaci, kterou chce přidat jako související organizaci

    10.4. Ulož id_organizace vybrané organizace do p.id_organizace
11. Zapiš hodnoty p.id_organizace, p.id_osoba do tabulky Osoba_Organizace jako nový záznam
12. END TRANSACTION
13. Jestliže se zápis neprovedl, vypiš chybovou hlášku a jdi na bod 4

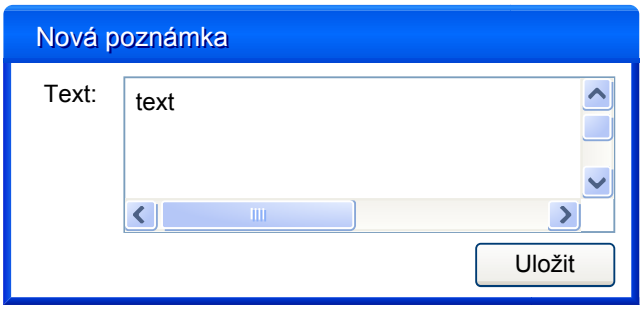
6.12 Přidání poznámky … funkce, která přidává do databáze IS textovou poznámku k vybrané osobě, resp. zákazníkovi.



Obrázek 3.44: DFD (6.12 Přidání poznámky)

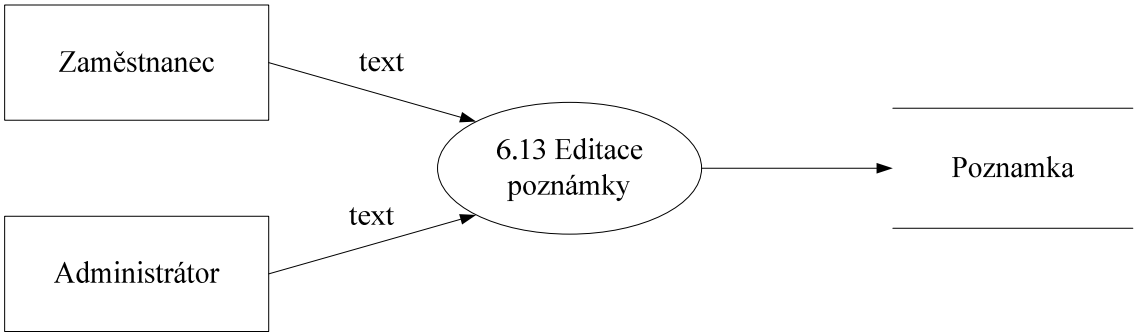
Algoritmus:

1. Zobraz formulář Osoba (viz. 6.10 Výpis osoby) s vybranou osobou, pro kterou chce uživatel přidat novou textovou poznámku
2. Ulož id_osoba vybrané osoby do p.id_osoba
3. Uživatel klikne na tlačítko pro přidání nové poznámky
4. Zobraz formulář Nová poznámka



5. Uživatel vyplní hodnotu text ve formuláři Nová poznámka
6. Ulož hodnotu text z formuláře Nová poznámka do p.text
7. Zkontroluj integritní omezení. Jestliže je chyba, vypiš chybovou hlášku a jdi na bod 4
8. Zapiš p.id_osoba, p.text do tabulky Poznamka jako nový záznam
9. Jestliže se zápis neprovedl, vypiš chybovou hlášku a jdi na bod 1

6.13 Editace poznámky … funkce, která dovoluje uživateli editovat textovou poznámku v databázi IS.

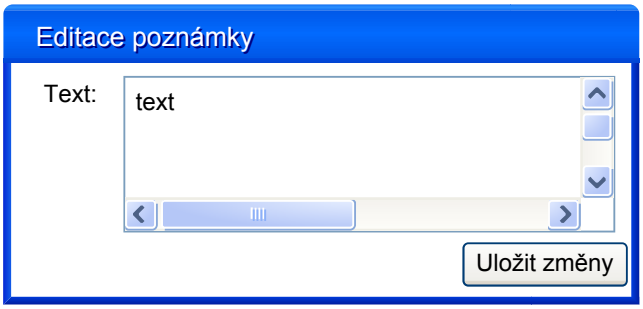


Obrázek 3.45: DFD (6.13 Editace poznámky)

Algoritmus:

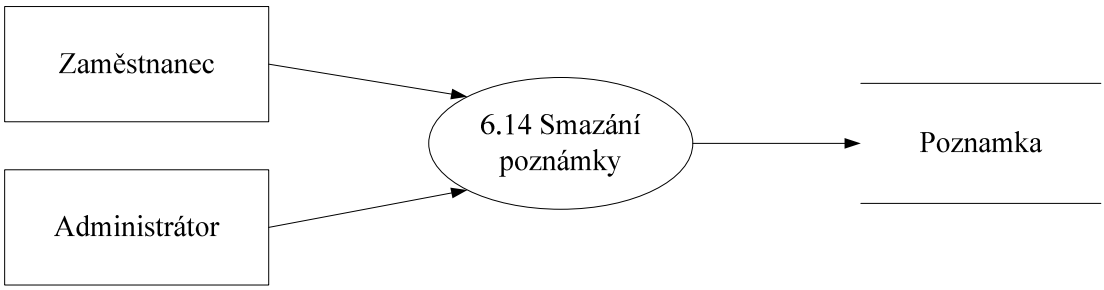
1. Zobraz formulář Přehled poznámek (viz. 6.15 Výpis poznámek)
2. Uživatel vybere poznámku, kterou chce editovat
3. Ulož id_poznamka vybrané poznámky do p.id_poznamka
4. Vyber z tabulky Poznamka záznam, kde Poznamka.id_poznamka = p.id_poznamka a ulož hodnoty do p.id_osoba, p.text

5. Zobraz formulář Editace poznámky



6. Uživatel modifikuje hodnotu ve formuláři Editace poznámky
7. Ulož hodnotu z formuláře Editace poznámky do p.text
8. Zkontroluj IO. Jestliže je chyba, vypiš chybovou hlášku a jdi na bod 5
9. Proveď zápis údajů p.id_osoba, p.text do tabulky Poznamka u záznamu, kde Poznamka.id_poznamka = p.id_poznamka
10. Jestliže se zápis neprovedl, vypiš chybovou hlášku a jdi na bod 1

6.14 Smazání poznámky … funkce, která odstraní poznámku z databáze IS.

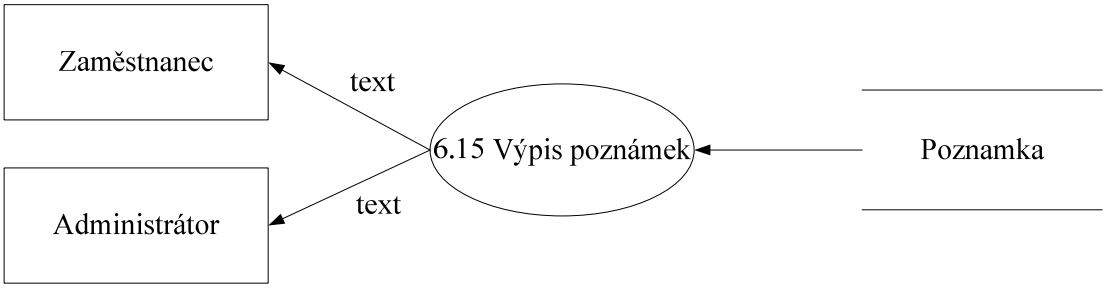


Obrázek 3.46: DFD (6.14 Smazání poznámky)

Algoritmus:

1. Zobraz formulář Přehled poznámek (viz. 6.15 Výpis poznámek)
2. Uživatel vybere poznámku, kterou chce smazat a klikne na tlačítko pro odstranění záznamu
3. Ulož id_poznamka vybrané poznámky do p.id_poznamka
4. Proveď odstranění záznamu z tabulky Poznamka, kde Poznamka.id_poznamka = p.id_poznamka
5. Jestliže se zápis neprovedl, vypiš chybovou hlášku a jdi na bod 1

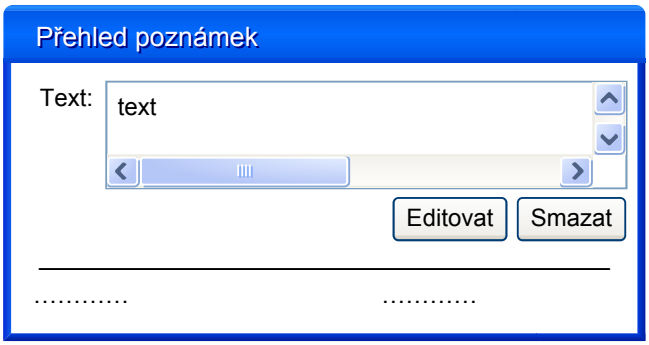
6.15 Výpis poznámek … funkce, která vypíše textové poznámky z databáze IS pro vybranou osobu.



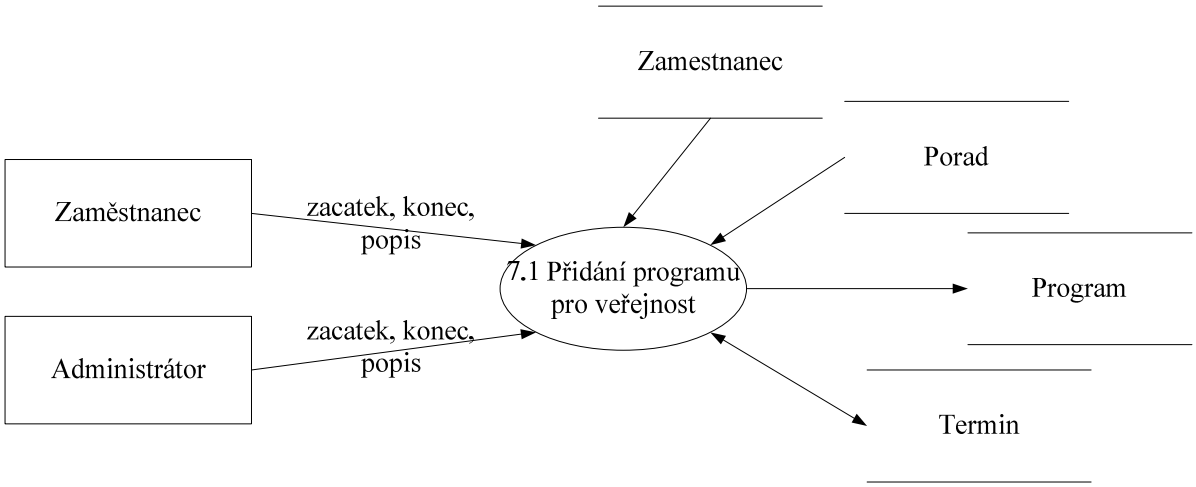
Obrázek 3.47: DFD (6.15 Výpis poznámek)

Algoritmus:

1. Zobraz formulář Osoba (viz. 6.10 Výpis osoby) s vybranou osobou, pro kterou chce uživatel zobrazit textové poznámky
2. Ulož id_osoba vybrané osoby do p.id_osoba
3. Uživatel klikne na tlačítko pro zobrazení textových poznámek
4. Načti z tabulky Poznamka všechny záznamy, kde Poznamka.id_osoba = p.id_osoba a ulož do p.id_osoba, p.text
5. Vypiš záznamy ve formuláři Přehled poznámek včetně odkazů pro editaci a smazání poznámky



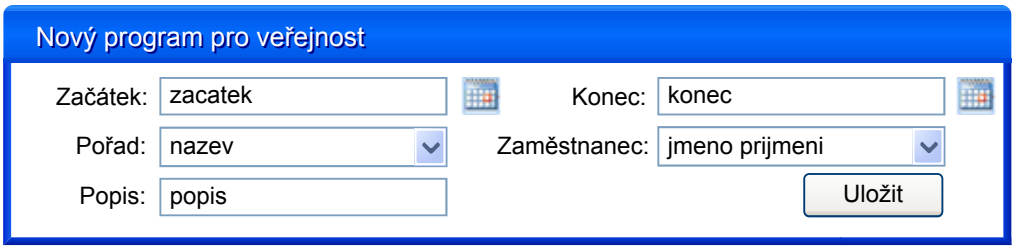
7.1 Přidání programu pro veřejnost … funkce, která přidává do databáze IS nový záznam o programu pro veřejnost. Program pro veřejnost přidává buď uživatelská role Administrátor, nebo Zaměstnanec na většinou pevně stanovený termín v týdnu. Na tento program pro veřejnost se potom mohou přidávat rezervace uživatelů.



Obrázek 3.48: DFD (7.1 Přidání programu pro veřejnost)

Algoritmus:

1. Zobraz formulář Nový program pro veřejnost

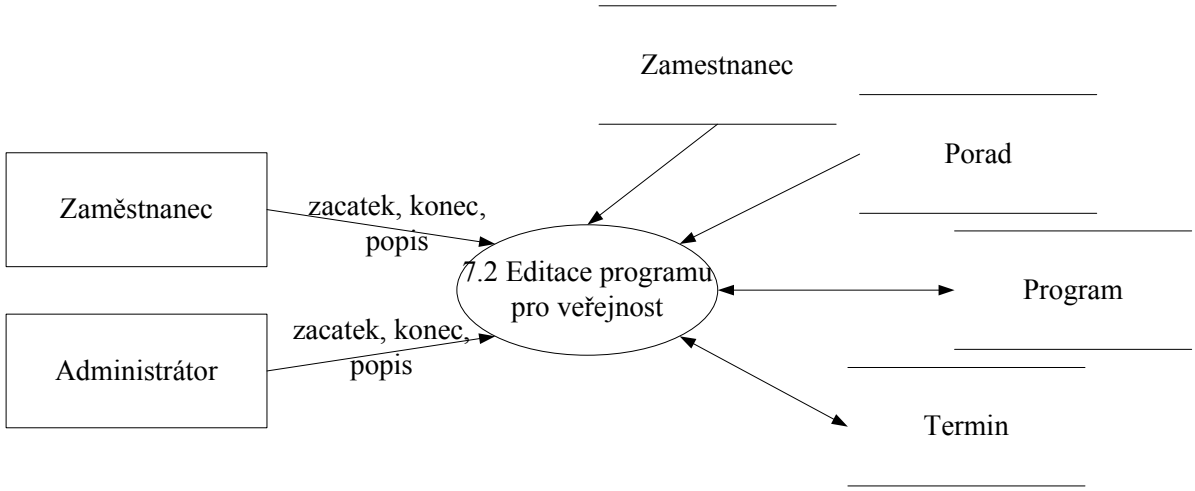


2. Uživatel vyplní formulář Nový program pro veřejnost
3. Ulož hodnoty z formuláře Nový program pro veřejnost do p.zacatek, p.konec, p.popis
4. Zkontroluj integritní omezení. Jestliže je chyba, vypiš chybovou hlášku a jdi na bod 1
5. Jestliže chce uživatel změnit zaměstnance
   PAK
   5.1. Zobraz seznam zaměstnanců z tabulky Zamestnanec
   5.2. Uživatel vybere zaměstnance podle jména a příjmení
   5.3. Ulož id_zamestnanec vybraného zaměstnance do p.id_zamestnanec
6. Jestliže chce uživatel změnit pořad
   PAK
   6.1. Zobraz seznam pořadů z tabulky Porad
   6.2. Uživatel vybere pořad podle názvu
   6.3. Ulož id_porad vybraného pořadu do p.id_porad
7. BEGIN TRANSACTION
8. Vyber Termin.id_termin z tabulky Termin, kde Termin.zacatek = p.zacatek AND Termin.konec = p.konec
9. Jestliže Termin.id_termin IS NOT NULL
   PAK
   9.1. Ulož Termin.id_termin do p.id_termin
   JINAK

9.2. Ulož hodnotu názvu dne v týdnu získané z hodnoty p.zacatek do p.den
9.3. Zapiš p.zacatek, p.konec, p.den, FALSE, FALSE do tabulky Termin jako nový záznam a vrať jeho id_termin do p.id_termin

10. Zapiš p.id_termin, p.id_porad, p.id_zamestnanec, p.popis do tabulky Program jako nový záznam
11. END TRANSACTION
12. Jestliže se zápis neprovedl, vypiš chybovou hlášku a jdi na bod 1

7.2 Editace programu pro veřejnost … funkce, která dovoluje uživateli editovat záznam o programu pro veřejnost v databázi IS.

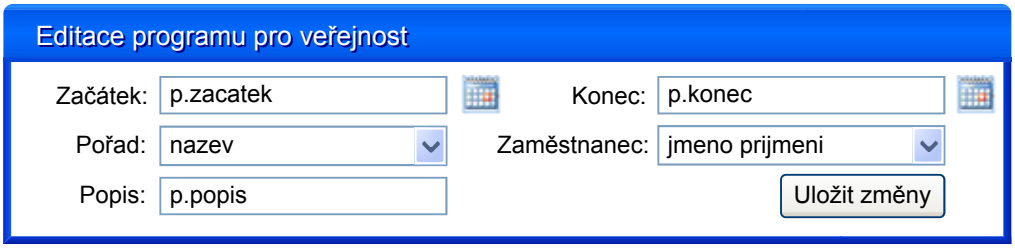


Obrázek 3.49: DFD (7.2 Editace programu pro veřejnost)

Algoritmus:

1. Zobraz formulář Program pro veřejnost (viz. 7.5 Výpis programu pro veřejnost) s vybraným programem pro veřejnost, který chce uživatel editovat
2. Ulož id_program vybraného programu pro veřejnost do p.id_program
3. Uživatel klikne na tlačítko pro editaci programu pro veřejnost
4. Načti z tabulky Program záznam, kde Program.id_program = p.id_program a ulož hodnoty do p.id_termin, p.id_porad, p.id_zamestnanec, p.popis
5. Doplň program pro veřejnost o atributy jmeno, prijmeni z tabulky Zamestnanec, kde Zamestnanec.id_zamestnanec = p.id_zamestnanec
6. Doplň program pro veřejnost o atribut nazev z tabulky Porad, kde Porad.id_porad = p.id_porad
7. Doplň objednávku o atributy zacatek, konec z tabulky Termin, kde Termin.id_termin = p.id_termin

8. Vypiš záznam doplněný o výše uvedené atributy ve formuláři Editace programu pro veřejnost



9. Uživatel modifikuje hodnoty ve formuláři Editace programu pro veřejnost
10. Ulož hodnoty z formuláře Editace programu pro veřejnost do p.zacatek, p.konec, p.popis
11. Zkontroluj integritní omezení. Jestliže je chyba, vypiš chybovou hlášku a jdi na bod 1
12. Jestliže chce uživatel změnit zaměstnance
    PAK
    12.1. Zobraz seznam zaměstnanců z tabulky Zamestnanec
    12.2. Uživatel vybere zaměstnance podle jména a příjmení
    12.3. Ulož id_zamestnanec vybraného zaměstnance do p.id_zamestnanec
13. Jestliže chce uživatel změnit pořad
    PAK
    13.1. Zobraz seznam pořadů z tabulky Porad
    13.2. Uživatel vybere pořad podle názvu
    13.3. Ulož id_porad vybraného pořadu do p.id_porad
14. BEGIN TRANSACTION
15. Vyber Termin.id_termin z tabulky Termin, kde Termin.zacatek = p.zacatek AND Termin.konec = p.konec
16. Jestliže Termin.id_termin IS NOT NULL
    PAK
    16.1. Ulož Termin.id_termin do p.id_termin
    JINAK
    16.2. Ulož hodnotu názvu dne v týdnu získané z hodnoty p.zacatek do p.den
    16.3. Proveď zápis údajů p.zacatek, p.konec, p.den do tabulky Termin, kde Termin.id_termin = p.id_termin a vrať jeho id_termin do p.id_termin
17. Proveď zápis údajů p.id_termin, p.id_porad, p.id_zamestnanec, p.id_popis do tabulky Program, kde Program.id_program = p.id_program
18. END TRANSACTION
19. Jestliže se zápis neprovedl, vypiš chybovou hlášku a jdi na bod 8
20. Jestliže selhalo cokoli jiného, vypiš chybovou hlášku a jdi na bod 1

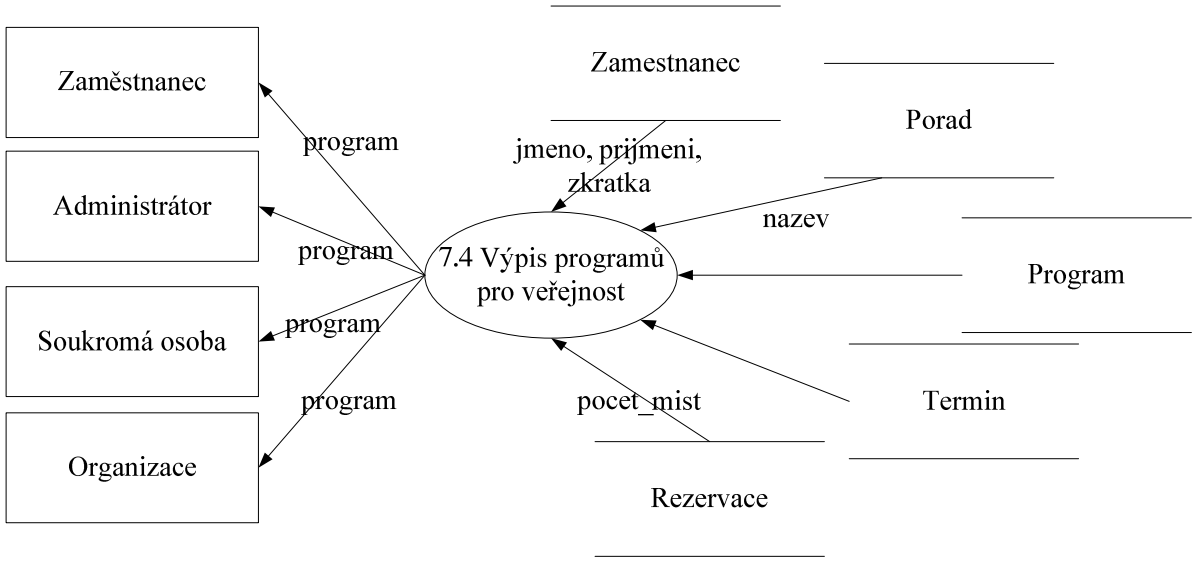
7.3 Smazání programu pro veřejnost … funkce, která odstraní vybraný program pro veřejnost z databáze IS.

Obrázek 3.50: DFD (7.3 Smazání programu pro veřejnost)

Algoritmus:

1. Zobraz formulář Program pro veřejnost (viz. 7.5 Výpis programu pro veřejnost) s vybraným programem pro veřejnost, který chce uživatel smazat
2. Ulož id_program vybraného programu pro veřejnost do p.id_program a id_termin do p.id_termin
3. Uživatel klikne na tlačítko pro odstranění programu pro veřejnost ze systému
4. BEGIN TRANSACTION
5. Jestliže neexistuje záznam v tabulce Program, kde Program.id_termin = p.id_termin AND Program.id_program != p.id_program
   PAK
   5.1. Proveď odstranění záznamu z tabulky Termin, kde Termin.id_termin = p.id_termin
6. Proveď odstranění záznamu z tabulky Program, kde Program.id_program = p.id_program
7. END TRANSACTION
8. Jestliže se zápis neprovedl, vypiš chybovou hlášku a jdi na bod 1

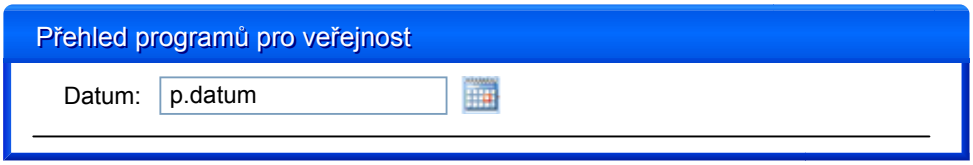
7.4 Výpis programů pro veřejnost … funkce poskytující přehledný výpis programů pro veřejnost v zadaném intervalu včetně celkového počtu rezervovaných míst na daný program pro veřejnost.



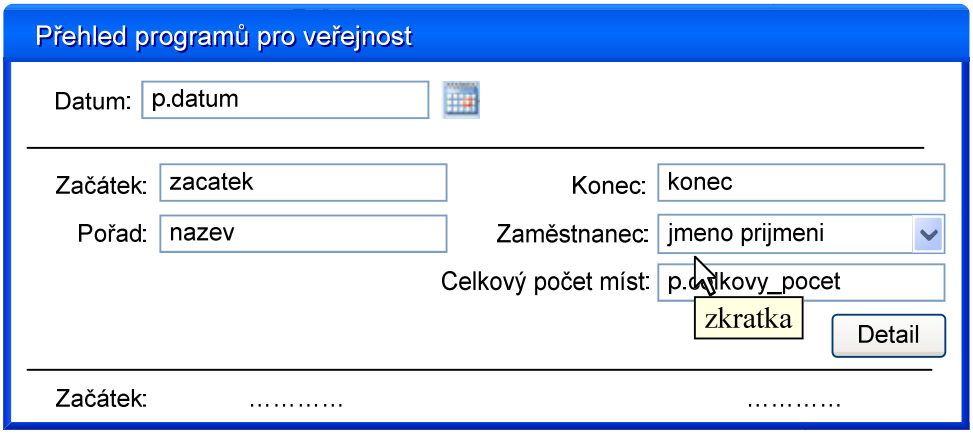
Obrázek 3.51: DFD (7.4 Výpis programů pro veřejnost)

Algoritmus:

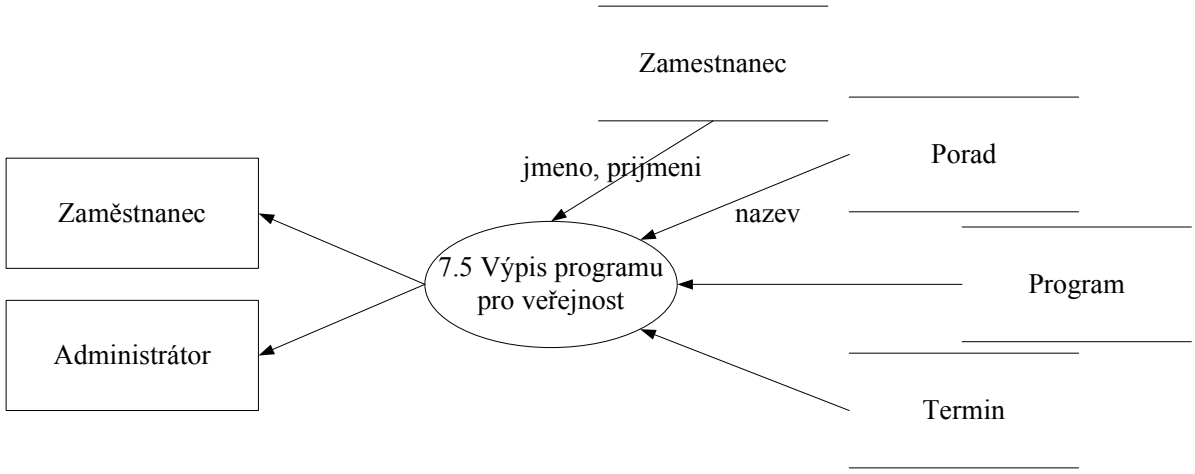
1. Zobraz záhlaví formuláře Přehled programů pro veřejnost obsahující hodnotu p.datum



2. Ulož do p.datum hodnotu aktuálního systémového data
3. Jestliže chce uživatel změnit hodnotu p.datum
   PAK
   3.1. Uživatel zadá do vstupního pole novou hodnotu položky datum (dd.MM.yyyy)
   3.2. Ulož datum do p.datum
4. Z hodnoty p.datum zjisti počátek týdne a konec týdne a ulož do p.zacatek a p.konec
5. Načti z tabulky Program všechny záznamy, kde Termin.zacatek <= p.zacatek AND Termin.konec <= p.konec AND Termin.id_termin = Program.id_termin a setřiď podle atributu Termin.zacatek
6. Doplň každý program pro veřejnost o atributy jmeno, prijmeni, zkratka z tabulky Zamestnanec, kde Zamestnanec.id_zamestnanec = Program.id_zamestnanec
7. Doplň každý program pro veřejnost o atribut nazev z tabulky Porad, kde Porad.id_porad = Program.id_porad
8. Doplň každý program pro veřejnost o atributy zacatek, konec z tabulky Termin, kde Termin.id_termin = Program.id_termin
9. Doplň každý program pro veřejnost o součet atributů pocet_mist z tabulky Rezervace, kde Rezervace.id_program = Program.id_program a ulož do p.celkovy_pocet
10. Vypiš záznamy doplněné o výše uvedené atributy ve formuláři Přehled programů pro veřejnost podle daného setřídění včetně odkazů pro detail programu pro veřejnost



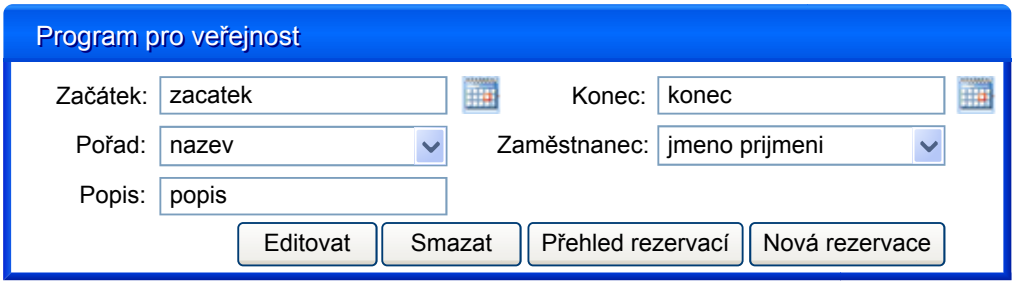
7.5 Výpis programu pro veřejnost … funkce, která vypíše detailní informace o vybraném programu pro veřejnost z databáze IS.



Obrázek 3.52: DFD (7.5 Výpis programu pro veřejnost)

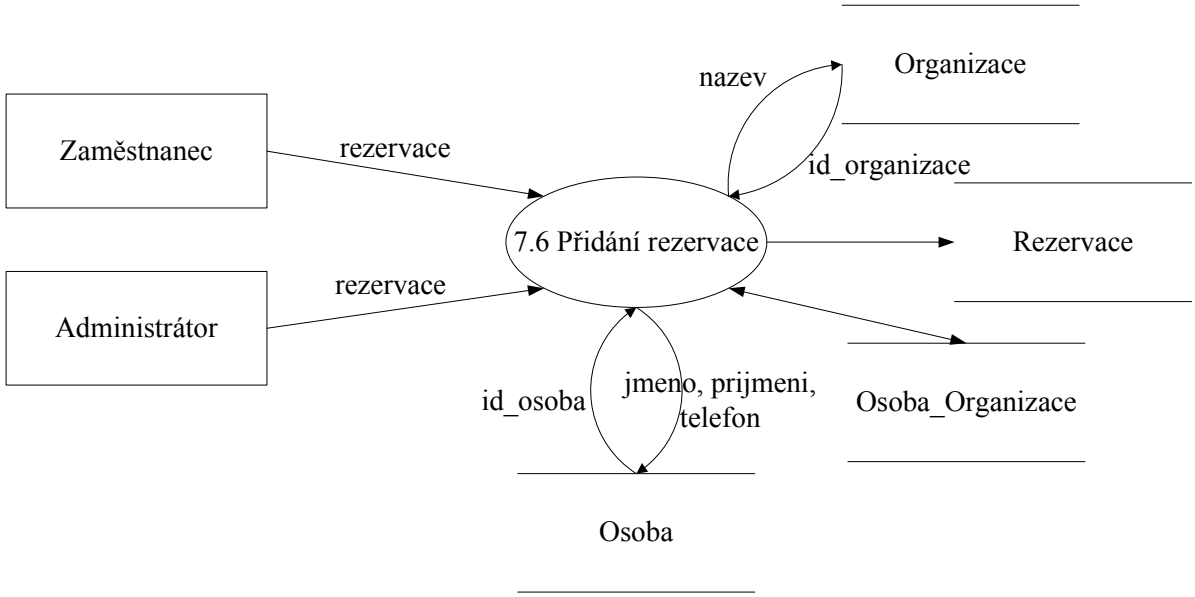
Algoritmus:

1. Zobraz formulář Přehled programů pro veřejnost (viz. 7.4 Výpis programů pro veřejnost)
2. Uživatel vybere program pro veřejnost, který chce detailně vypsat
3. Ulož id_program vybraného programu pro veřejnost do p.id_program
4. Načti z tabulky Program záznam, kde Program.id_program = p.id_program a ulož hodnoty do p.id_termin, p.id_porad, p.id_zamestnanec, p.popis
5. Doplň program pro veřejnost o atributy jmeno, prijmeni z tabulky Zamestnanec, kde Zamestnanec.id_zamestnanec = p.id_zamestnanec
6. Doplň program pro veřejnost o atribut nazev z tabulky Porad, kde Porad.id_porad = p.id_porad
7. Doplň objednávku o atributy zacatek, konec z tabulky Termin, kde Termin.id_termin = p.id_termin
8. Vypiš záznam doplněný o výše uvedené atributy ve formuláři Program pro veřejnost včetně odkazů pro další funkce spojené s vybraným programem pro veřejnost



7.6 Přidání rezervace … funkce, která přidává do databáze IS novou rezervaci na vybraný program pro veřejnost. Tuto funkci mohou v systému použít všechny role, ale minispecifikace bude odlišná podle toho, jestli jde o roli zaměstnance (Administrátor, Zaměstnanec) nebo zákazníka (Organizace, Soukromá osoba). Popíšu zde minispecifikaci z pohledu zaměstnanců HaPJP, neboť ta je komplexnější a popisuje více detailů. Přidání nové rezervace pro zákazníky systému probíhá v podstatě pouze tak, že si uživatel vybere program pro veřejnost a zadá počet míst. Provede se kontrola IO, a pokud je vše v pořádku, tak je nový záznam o rezervaci uložen do databáze IS a

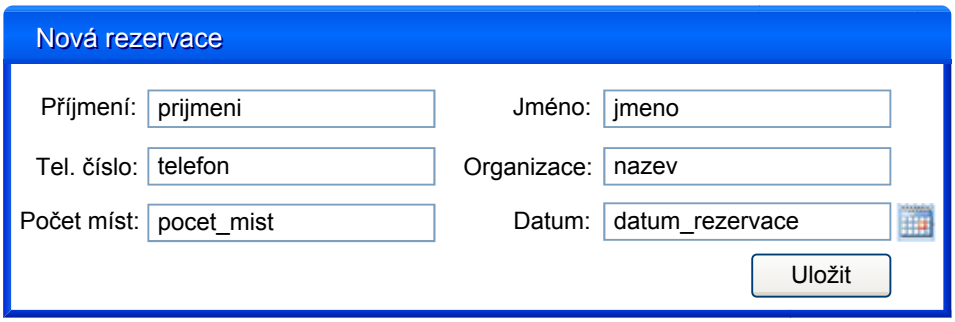
vygeneruje se automaticky email, který přijde na emailovou adresu nastavenou v konfiguračním souboru.



Obrázek 3.53: DFD (7.6 Přidání rezervace)

Algoritmus:

10. Zobraz formulář Program pro veřejnost (viz. 7.5 Výpis programu pro veřejnost) s vybraným programem pro veřejnost, na který chce uživatel přidat novou rezervaci
11. Ulož id_program vybraného programu pro veřejnost do p.id_program
12. Uživatel klikne na tlačítko pro přidání nové rezervace
13. Zobraz formulář Nová rezervace

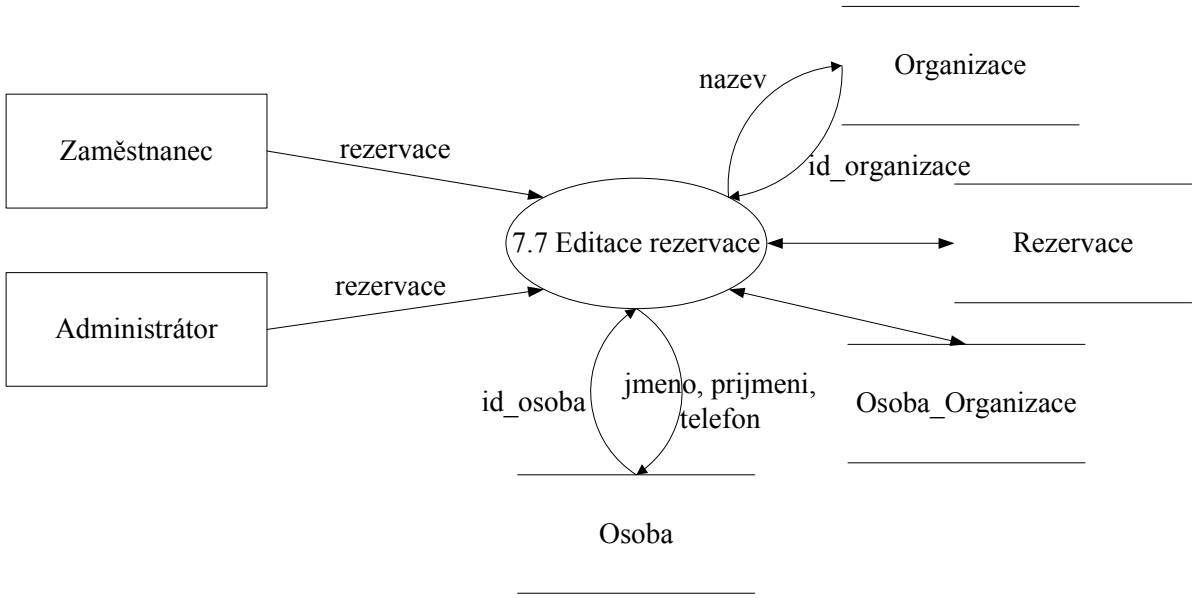


14. Uživatel vyplní hodnoty na formuláři Nová rezervace
15. Ulož hodnoty z formuláře Nová rezervace do p.jmeno, p.prijmeni, p.telefon, p. nazev_organizace, p.pocet_mist, p.datum_rezervace
16. Zkontroluj integritní omezení. Jestliže je chyba, vypiš chybovou hlášku a jdi na bod 4
17. BEGIN TRANSACTION
18. Vyber Organizace.id_organizace z tabulky Organizace, kde Organizace.nazev = p.nazev_organizace
19. Jestliže Organizace.id_organizace IS NOT NULL
    PAK

10.1. Ulož atribut Organizace.id_organizace do p.id_organizace
JINAK
10.2. Ulož p.nazev_organizace do p.nazev
10.3. Zapiš p.nazev do tabulky Organizace jako nový záznam a vrať jeho id_organizace do p.id_organizace

20. Vyber Osoba.id_osoba z tabulky Osoba, kde Osoba.jmeno = p.jmeno AND Osoba.prijmeni = p.prijmeni AND Osoba.telefon = p.telefon
21. Jestliže Osoba.id_osoba IS NOT NULL
PAK
12.1. Ulož Osoba.id_osoba do p.id_osoba
JINAK
12.2. Vyber Osoba.id_osoba z tabulky Osoba, kde Osoba.jmeno = p.jmeno AND Osoba.prijmeni = p.prijmeni
12.3. PRO KAŽDÉ Osoba.id_osoba DĚLEJ
12.3.1. Jestliže existuje záznam v tabulce Osoba_Organizace, kde Osoba_Organizace.id_osoba = Osoba.id_osoba AND Osoba_Organizace.id_organizace = p.id_organizace
PAK
12.3.1.1. Ulož Osoba.id_osoba do p.id_osoba
12.3.1.2. Zapiš p.telefon do tabulky Osoba, kde Osoba.id_osoba = p.id_osoba
12.4. Jestliže neexistuje žádný záznam v tabulce Osoba, kde Osoba.jmeno = p.jmeno AND Osoba.prijmeni = p.prijmeni nebo p.id_osoba IS NULL
PAK
12.4.1. Zapiš p.jmeno, p.prijmeni, p.telefon do tabulky Osoba jako nový záznam a vrať jeho id_osoba do p.id_osoba
22. Jestliže neexistuje záznam v tabulce Osoba_Organizace, kde Osoba_Organizace.id_osoba = p.id_osoba AND Osoba_Organizace.id_organizace = p.id_organizace
PAK
13.1. Zapiš p.id_organizace, p.id_osoba do tabulky Osoba_Organizace jako nový záznam
23. END TRANSACTION
24. Jestliže se zápis neprovedl, vypiš chybovou hlášku a jdi na bod 1
25. Zapiš p.id_program, p.id_osoba, p.id_organizace, p.pocet_mist, p.datum_rezervace do tabulky Rezervace jako nový záznam
26. Jestliže se zápis neprovedl, vypiš chybovou hlášku a jdi na bod 1

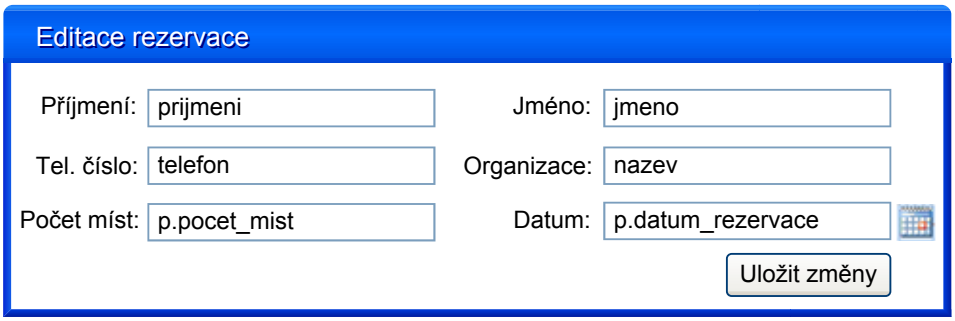
7.7 Editace rezervace … funkce, která dovoluje uživateli změnit údaje o rezervaci v databázi IS.



Obrázek 3.54: DFD (7.7 Editace rezervace)

Algoritmus:

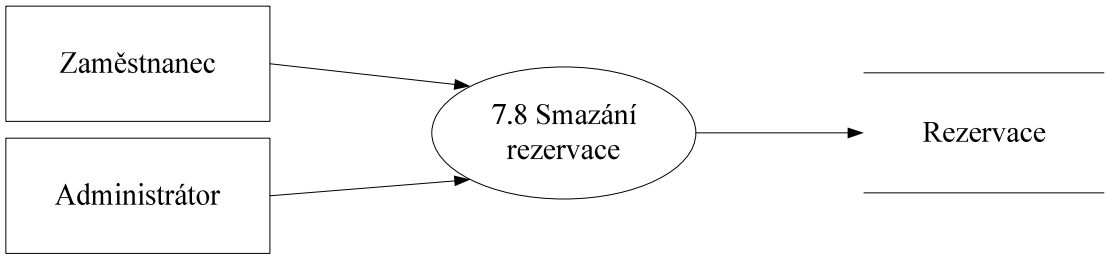
1. Zobraz formulář Přehled rezervací (viz. 7.9 Výpis rezervací)
2. Uživatel vybere rezervaci, kterou chce editovat
3. Ulož id_rezervace vybrané rezervace do p.id_rezervace
4. Načti z tabulky Rezervace záznam, kde Rezervace.id_rezervace = p.id_rezervace a ulož hodnoty do p.id_program, p.id_osoba, p.id_organizace, p.pocet_mist, p.datum_rezervace
5. Doplň záznam o atributy jmeno, prijmeni, telefon z tabulky Osoba, kde Osoba.id_osoba = p.id_osoba
6. Doplň záznam o atribut nazev z tabulky Organizace, kde Organizace.id_organizace = p.id_organizace
7. Vypiš záznam doplněný o výše uvedené atributy ve formuláři Editace rezervace



8. Uživatel modifikuje hodnoty ve formuláři Editace rezervace
9. Ulož hodnoty z formuláře Editace rezervace do p.jmeno, p.prijmeni, p.telefon, p.nazev_organizace, p.pocet_mist, p.datum_rezervace
10. Zkontroluj integritní omezení. Jestliže je chyba, vypiš chybovou hlášku a jdi na bod 7
11. BEGIN TRANSACTION
12. Vyber Organizace.id_organizace z tabulky Organizace, kde Organizace.nazev = p.nazev_organizace

13. Jestliže Organizace.id_organizace IS NOT NULL
    PAK
    13.1. Ulož atribut Organizace.id_organizace do p.id_organizace
    JINAK
    13.2. Ulož p.nazev_organizace do p.nazev
    13.3. Zapiš p.nazev do tabulky Organizace jako nový záznam a vrať jeho id_organizace do p.id_organizace
14. Vyber Osoba.id_osoba z tabulky Osoba, kde Osoba.jmeno = p.jmeno AND Osoba.prijmeni = p.prijmeni AND Osoba.telefon = p.telefon
15. Jestliže Osoba.id_osoba IS NOT NULL
    PAK
    15.1. Ulož Osoba.id_osoba do p.id_osoba
    JINAK
    15.2. Vyber Osoba.id_osoba z tabulky Osoba, kde Osoba.jmeno = p.jmeno AND Osoba.prijmeni = p.prijmeni
    15.3. PRO KAŽDÉ Osoba.id_osoba DĚLEJ
        15.3.1. Jestliže existuje záznam v tabulce Osoba_Organizace, kde Osoba_Organizace.id_osoba = Osoba.id_osoba AND Osoba_Organizace.id_organizace = p.id_organizace
            PAK
            15.3.1.1. Ulož Osoba.id_osoba do p.id_osoba
            15.3.1.2. Zapiš p.telefon do tabulky Osoba, kde Osoba.id_osoba = p.id_osoba
    15.4. Jestliže neexistuje žádný záznam v tabulce Osoba, kde Osoba.jmeno = p.jmeno AND Osoba.prijmeni = p.prijmeni nebo p.id_osoba IS NULL
        PAK
        15.4.1. Zapiš p.jmeno, p.prijmeni, p.telefon do tabulky Osoba jako nový záznam a vrať jeho id_osoba do p.id_osoba
16. Jestliže neexistuje záznam v tabulce Osoba_Organizace, kde Osoba_Organizace.id_osoba = p.id_osoba AND Osoba_Organizace.id_organizace = p.id_organizace
    PAK
    16.1. Zapiš p.id_organizace, p.id_osoba do tabulky Osoba_Organizace jako nový záznam
17. END TRANSACTION
18. Jestliže se zápis neprovedl, vypiš chybovou hlášku a jdi na bod 1
19. Proveď zápis údajů p.id_program, p.id_osoba, p.id_organizace, p.pocet_mist, p.datum_rezervace do tabulky Rezervace, kde Rezervace.id_rezervace = p.id_rezervace
20. Jestliže se zápis neprovedl, vypiš chybovou hlášku a jdi na bod 1

7.8 Smazání rezervace … funkce, která odstraní záznam o rezervaci z databáze IS.

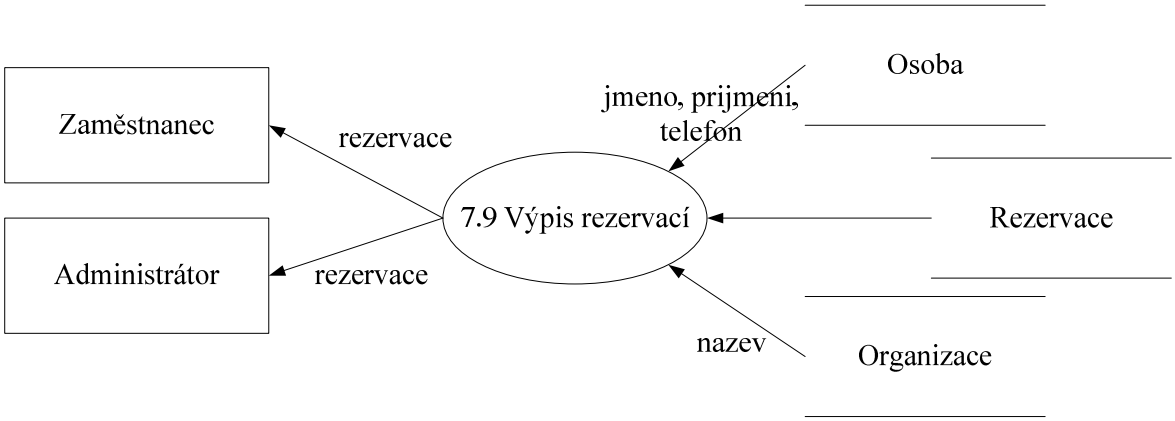


Obrázek 3.55: DFD (7.8 Smazání rezervace)

Algoritmus:

1. Zobraz formulář Přehled rezervací (viz. 7.9 Výpis rezervací)
2. Uživatel vybere rezervaci, kterou chce smazat
3. Ulož id_rezervace vybrané rezervace do p.id_rezervace
4. Proveď odstranění záznamu z tabulky Rezervace, kde Rezervace.id_rezervace = p.id_rezervace
5. Jestliže se zápis neprovedl, vypiš chybovou hlášku a jdi na bod 1

7.9 Výpis rezervací … funkce, která poskytuje přehledný výpis všech rezervací na vybraný program pro veřejnost.

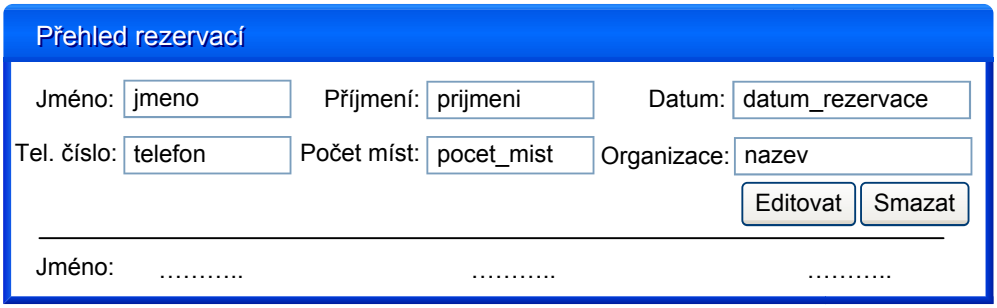


Obrázek 3.56: DFD (7.9 Výpis rezervací)

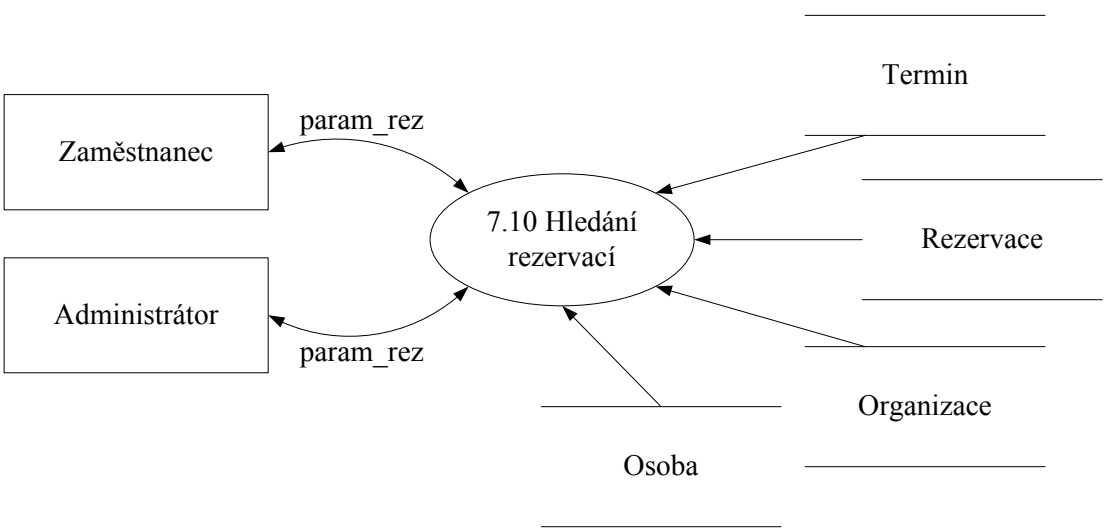
Algoritmus:

1. Zobraz formulář Program pro veřejnost (viz. 7.5 Výpis programu pro veřejnost) s vybraným programem pro veřejnost, ke kterému chce uživatel zobrazit všechny rezervace
2. Ulož id_program vybraného programu pro veřejnost do p.id_program
3. Uživatel klikne na tlačítko pro zobrazení všech rezervací
4. Načti všechny záznamy z tabulky Rezervace, kde Rezervace.id_program = p.id_program a ulož hodnoty do p.id_osoba, p.id_organizace, p.pocet_mist, p.datum_rezervace a setřiď podle atributu datum_rezervace
5. Doplň každý záznam o atributy jmeno, prijmeni, telefon z tabulky Osoba, kde Osoba.id_osoba = p.id_osoba
6. Doplň každý záznam o atribut nazev z tabulky Organizace, kde Organizace.id_organizace = p.id_organizace

7. Vypiš všechny záznamy doplněné o výše uvedené atributy ve formuláři Přehled rezervací podle daného setřídění a včetně odkazů pro smazání a editaci rezervace



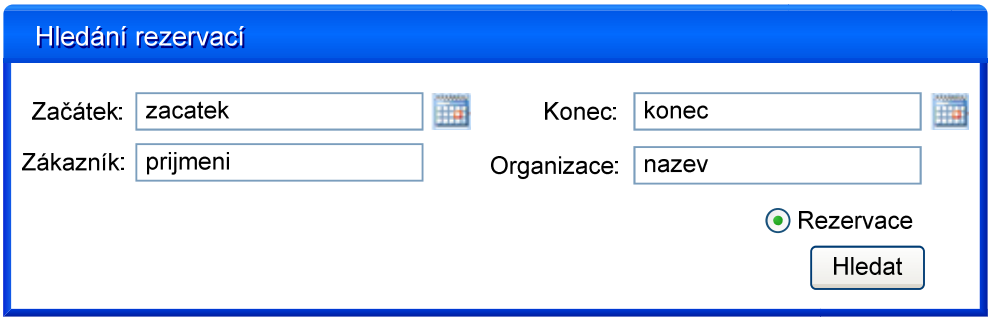
7.10 Hledání rezervací … funkce, která poskytuje možnost vyhledat rezervace v databázi systému podle předem stanovených kritérií.



Obrázek 3.57: DFD (7.10 Hledání rezervací)

Algoritmus:

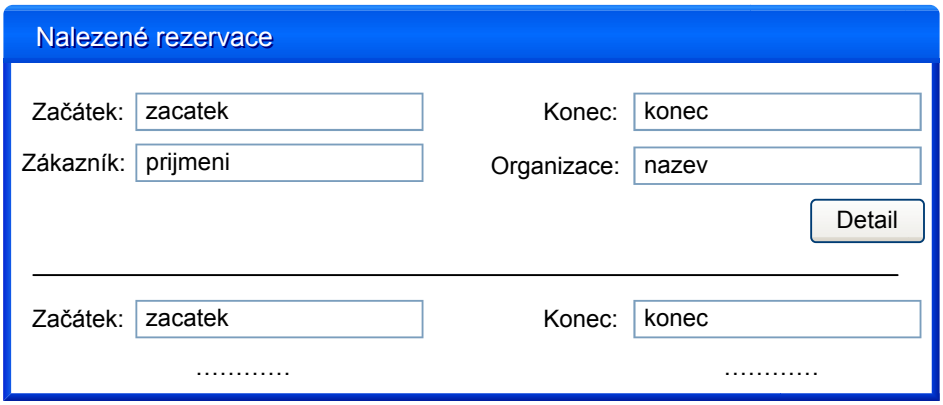
1. Zobraz formulář Hledání rezervací



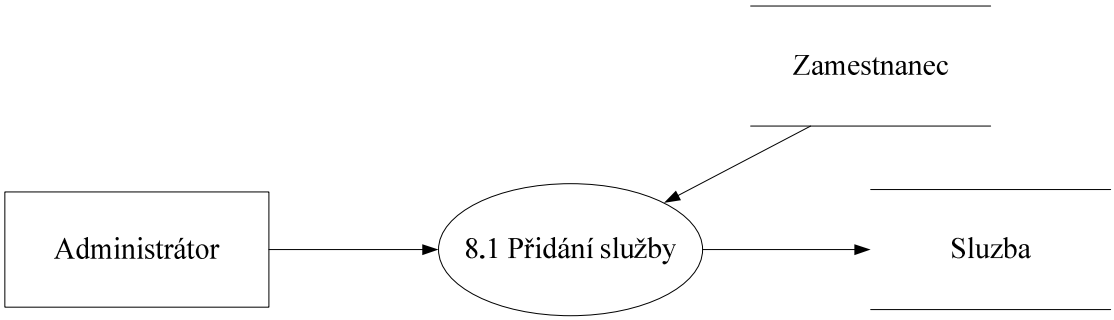
2. Uživatel modifikuje vstupní parametry ve formuláři Hledání rezervací
3. Ulož hodnoty z formuláře Hledání rezervací do p.zacatek, p.konec, p.prijmeni, p.nazev
4. Uživatel klikne na tlačítko pro vyhledání rezervací podle zadaných parametrů
5. Načti z tabulky Rezervace všechny záznamy, kde Termin.zacatek <= p.zacatek AND Termin.konec <= p.konec AND Termin.id_termin = Program.id_termin AND

Program.id_program = Rezervace.id_program AND Osoba.prijmeni LIKE p.prijmeni AND Osoba.id_osoba = Rezervace.id_osoba AND Organizace.nazev LIKE p.nazev AND Organizace.id_organizace = Rezervace.id_organizace a ulož hodnoty do p.zacatek, p.konec, p.prijmeni, p.nazev

6. Vypiš všechny nalezené záznamy do formuláře Nalezené rezervace a zobraz tlačítko pro přechod na detail rezervace, resp. na program pro veřejnost



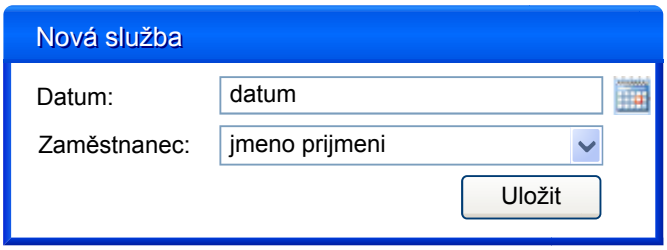
8.1 Přidání služby … funkce, která přidává do databáze IS záznam o službě zaměstnance.



Obrázek 3.58: DFD (8.1 Přidání služby)

Algoritmus:

1. Zobraz formulář Nová služba

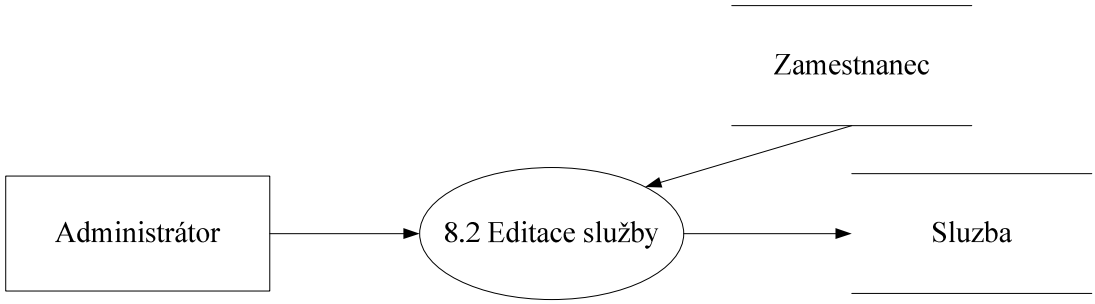


2. Uživatel vyplní formulář Nová služba
3. Ulož hodnotu datum z formuláře Nová služba do p.datum
4. Zkontroluj integritní omezení. Jestliže je chyba, vypiš chybovou hlášku a jdi na bod 1
5. Jestliže chce uživatel změnit ID zaměstnance
   PAK
   5.1. Zobraz seznam zaměstnanců z tabulky Zamestnanec

5.2. Uživatel vybere zaměstnance
5.3. Ulož id_zamestnanec vybraného zaměstnance do p.id_zamestnanec

6. Zapiš p.datum, p.id_zamestnanec jako nový záznam do tabulky Sluzba
7. Jestliže se zápis neprovedl, vypiš chybovou hlášku a jdi na bod 1

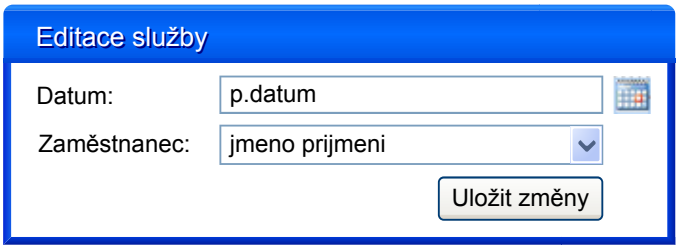
8.2 Editace služby … funkce pro editaci služby zaměstnance v databázi IS.



Obrázek 3.59: DFD (8.2 Editace služby)

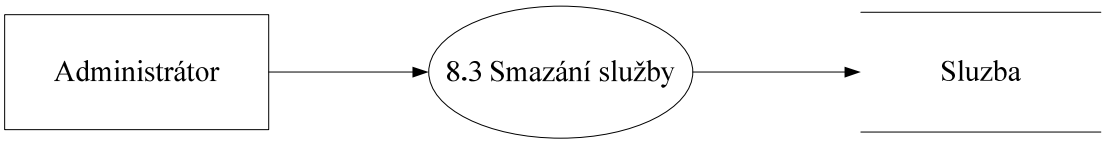
Algoritmus:

11. Zobraz formulář Přehled služeb (viz. 8.4 Výpis služeb)
12. Uživatel vybere službu, kterou chce editovat
13. Ulož id_sluzba vybrané služby do p.id_sluzba
14. Vyber z tabulky Sluzba záznam, kde Sluzba.id_sluzba = p.id_sluzba a ulož hodnoty do p.id_zamestnanec, p.datum
15. Doplň službu o atributy jmeno, prijmeni z tabulky Zamestnanec, kde Zamestnanec.id_zamestnanec = p.id_zamestnanec
16. Zobraz formulář Editace služby



17. Uživatel modifikuje hodnoty ve formuláři Editace služby
18. Ulož hodnotu z formuláře Editace služby do p.datum
19. Zkontroluj IO. Jestliže je chyba, vypiš chybovou hlášku a jdi na bod 6
20. Jestliže chce uživatel změnit ID zaměstnance
    PAK
    10.1. Zobraz seznam zaměstnanců z tabulky Zamestnanec
    10.2. Uživatel vybere zaměstnance
    10.3. Ulož id_zamestnanec vybraného zaměstnance do p.id_zamestnanec
21. Proveď zápis údajů p.datum, p.id_zamestnanec do tabulky Sluzba u záznamu, kde Sluzba.id_sluzba = p.id_sluzba
22. Jestliže se zápis neprovedl, vypiš chybovou hlášku a jdi na bod 1

8.3 Smazání služby … funkce, která odstraní službu zaměstnance z databáze IS.

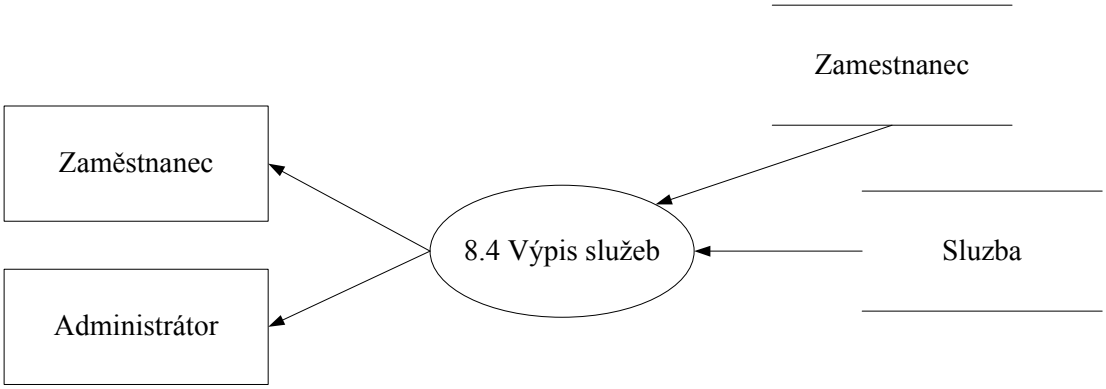


Obrázek 3.60: DFD (8.3 Smazání služby)

Algoritmus:

6. Zobraz formulář Přehled služeb (viz. 8.4 Výpis služeb)
7. Uživatel vybere službu, kterou chce smazat a klikne na tlačítko pro odstranění záznamu
8. Ulož id_sluzba vybrané služby do p.id_sluzba
9. Proveď odstranění záznamu z tabulky Sluzba, kde Sluzba.id_sluzba = p.id_sluzba
10. Jestliže se zápis neprovedl, vypiš chybovou hlášku a jdi na bod 1

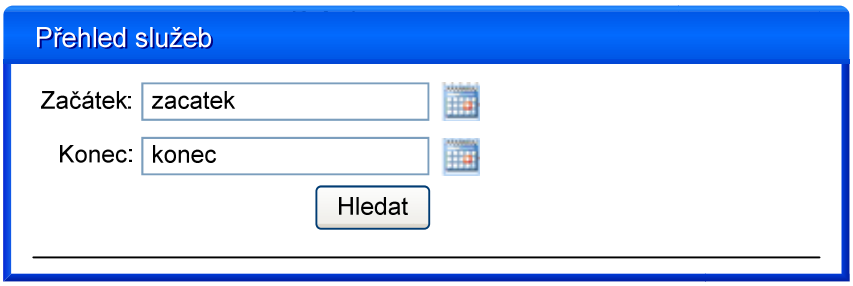
8.4 Výpis služeb … funkce, která vypíše služby z databáze do formuláře Přehled služeb.



Obrázek 3.61: DFD (8.4 Výpis služeb)

Algoritmus:

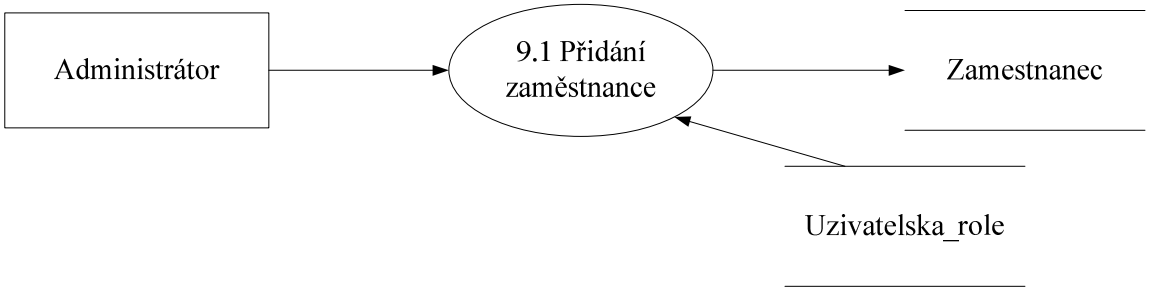
1. Zobraz záhlaví formuláře Přehled služeb



2. Uživatel vyplní hodnoty zacatek a konec
3. Ulož hodnoty z formuláře Přehled služeb do p.zacatek a p.konec
4. Zkontroluj IO. Jestliže je chyba, vypiš chybovou hlášku a jdi na bod 1
5. Uživatel klikne na tlačítko Hledat
6. Načti z tabulky Sluzba všechny záznamy, kde Sluzba.datum >= p.zacatek AND Sluzba.datum <= p.konec a setřiď je podle atributu datum
7. Doplň každou službu o atributy jmeno, prijmeni z tabulky Zamestnanec, kde Zamestnanec.id_zamestnanec = Sluzba.id_zamestnanec

8. Vypiš záznamy ve formuláři Přehled služeb podle daného setřídění včetně odkazů pro editaci a smazání služby

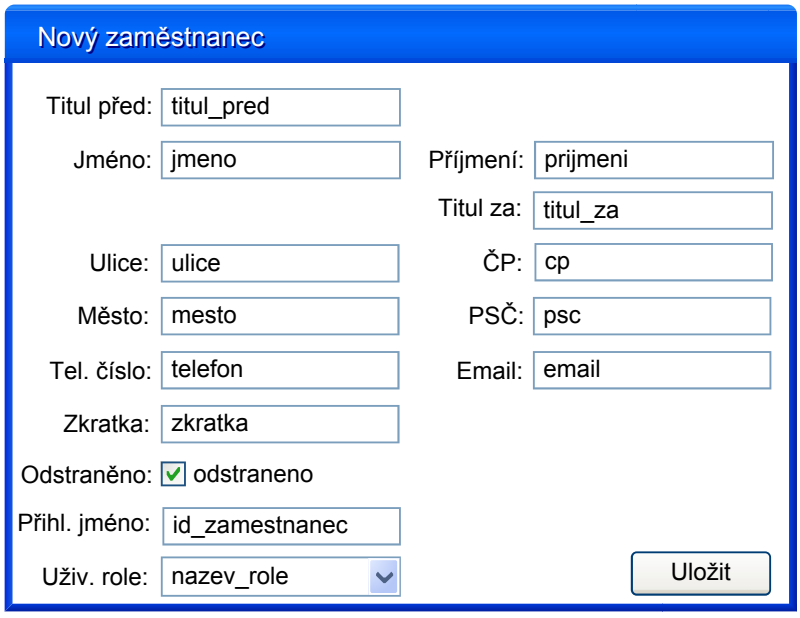
9.1 Přidání zaměstnance … funkce, která přidává nový záznam o zaměstnanci do databáze IS.



Obrázek 3.62: DFD (9.1 Přidání zaměstnance)

Algoritmus:

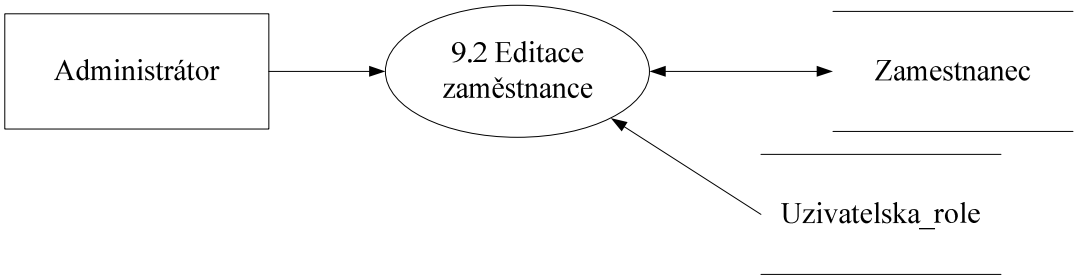
1. Zobraz formulář Nový zaměstnanec



2. Uživatel vyplní formulář Nový zaměstnanec

3. Ulož hodnoty z formuláře Nový zaměstnanec do p.titul_pred, p.jmeno, p.prijmeni, p.titul_za, p.ulice, p.cp, p.mesto, p.psc, p.telefon, p.email, p.zkratka, p.odstraneno, p.id_zamestnanec
4. Zkontroluj integritní omezení. Jestliže je chyba, vypiš chybovou hlášku a jdi na bod 1
5. Jestliže chce uživatel změnit ID role pro nového zaměstnance
   PAK
   5.1. Zobraz seznam uživatelských rolí z tabulky Uzivatelska_role
   5.2. Uživatel vybere roli z nabídky podle atributu nazev_role
   5.3. Ulož id_role vybrané role do p.id_role
6. Zapiš p.id_zamestnanec, p.id_role, p.titul_pred, p.jmeno, p.prijmeni, p.titul_za, p.ulice, p.cp, p.mesto, p.psc, p.telefon, p.email, p.zkratka, p.odstraneno do tabulky Zamestnanec jako nový záznam
7. Jestliže se zápis neprovedl, vypiš chybovou hlášku a jdi na bod 1

9.2 Editace zaměstnance … funkce, která dovoluje uživateli v roli Administrátora editovat záznam o zaměstnanci v databázi IS.

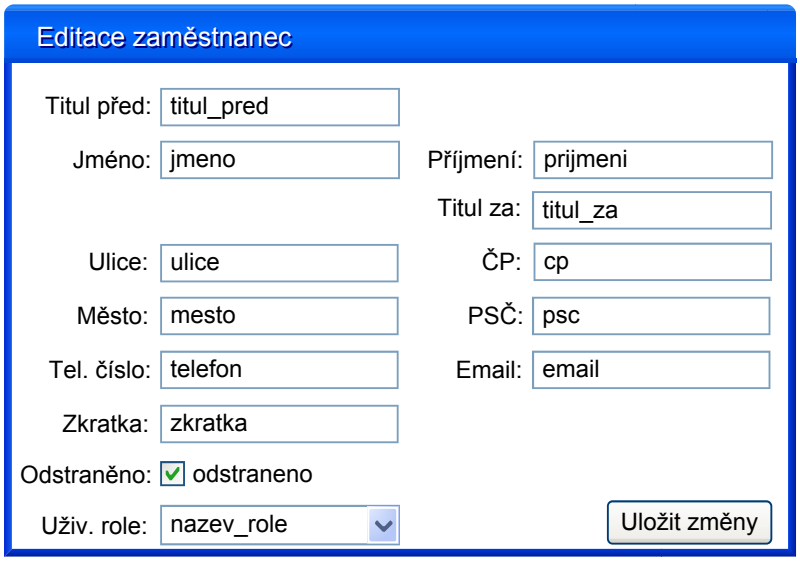


Obrázek 3.63: DFD (9.2 Editace zaměstnance)

Algoritmus:

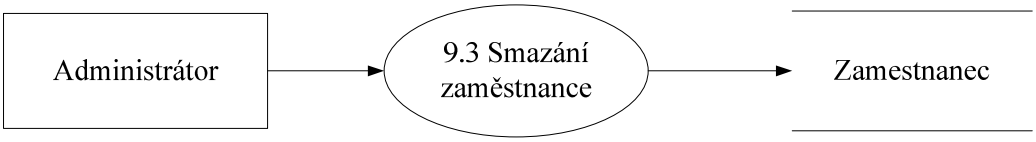
1. Zobraz formulář Zaměstnanec (viz. 9.5 Výpis zaměstnance) s vybraným zaměstnancem, kterého chce uživatel editovat
2. Ulož id_zamestnance vybraného zaměstnance do p.id_zamestnanec
3. Uživatel klikne na tlačítko pro editaci zaměstnance
4. Načti z tabulky Zamestnanec záznam, kde Zamestnanec.id_zamestnanec = p.id_zamestnanec a ulož hodnoty do p.id_role, p.titul_pred, p.jmeno, p.prijmeni, p.titul_za, p.ulice, p.cp, p.mesto, p.psc, p.telefon, p.email, p.zkratka, p.odstraneno
5. Doplň zaměstnance o atribut nazev_role z tabulky Uzivatelska_role, kde Uzivatelska_role.id_role = p.id_role

6. Vypiš záznam doplněný o výše uvedený atribut ve formuláři Editace zaměstnance



7. Uživatel modifikuje hodnoty ve formuláři Editace zaměstnance
8. Ulož hodnoty z formuláře Editace zaměstnance do p.titul_pred, p.jmeno, p.prijmeni, p.titul_za, p.ulice, p.cp, p.mesto, p.psc, p.telefon, p.email, p.zkratka, p.odstraneno
9. Zkontroluj IO. Jestliže je chyba, vypiš chybovou hlášku a jdi na bod 6
10. Jestliže chce uživatel změnit ID uživatelské role pro editovaného zaměstnance
    PAK
    10.1. Zobraz seznam uživatelských rolí z tabulky Uzivatelska_role
    10.2. Uživatel vybere uživatelskou roli
    10.3. Ulož id_role vybrané uživatelské role do p.id_role
11. Proveď zápis údajů p.id_role, p.titul_pred, p.jmeno, p.prijmeni, p.titul_za, p.ulice, p.cp, p.mesto, p.psc, p.telefon, p.email, p.zkratka, p.odstraneno do tabulky Zamestnanec u záznamu, kde Zamestnanec.id_zamestnanec = p.id_zamestnanec
12. Jestliže se zápis neprovedl, vypiš chybovou hlášku a jdi na bod 1

9.3 Smazání zaměstnance … funkce, která smaže záznam o zaměstnanci z databáze IS.



Obrázek 3.64: DFD (9.3 Smazání zaměstnance)

Algoritmus:

9. Zobraz formulář Zaměstnanec (viz. 9.5 Výpis zaměstnance) s vybraným zaměstnancem, kterého chce uživatel smazat
10. Ulož id_zamestnanec vybraného zaměstnance do p.id_zamestnanec
11. Uživatel klikne na tlačítko pro smazání zaměstnance
12. Proveď odstranění záznamu z tabulky Zamestnanec, kde Zamestnanec.id_zamestnanec = p.id_zamestnanec

13. Jestliže se zápis neprovedl, vypiš chybovou hlášku a jdi na bod 1

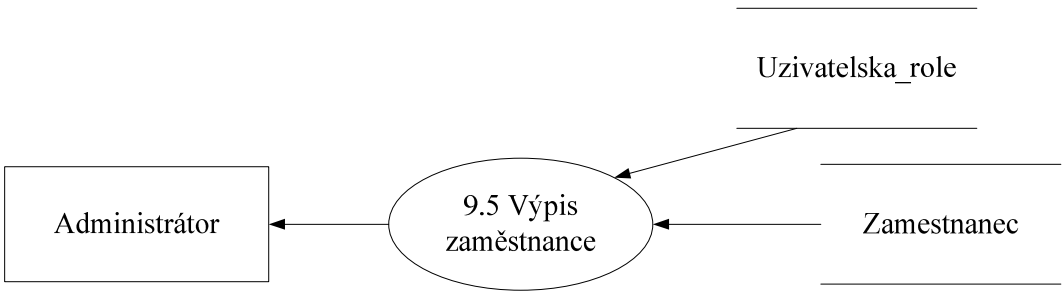
9.4 Výpis zaměstnanců … funkce, která poskytuje přehledný výpis všech zaměstnanců.

Obrázek 3.65: DFD (9.4 Výpis zaměstnanců)

Algoritmus:

7. Načti z tabulky Zamestnanec všechny záznamy a setřiď podle atributu jmeno, prameni
8. Vypiš záznamy ve formuláři Přehled zaměstnanců podle daného setřídění včetně odkazů pro detail zaměstnance

9.5 Výpis zaměstnance … funkce, která poskytuje výpis detailních informací o vybraném zaměstnanci z databáze IS.



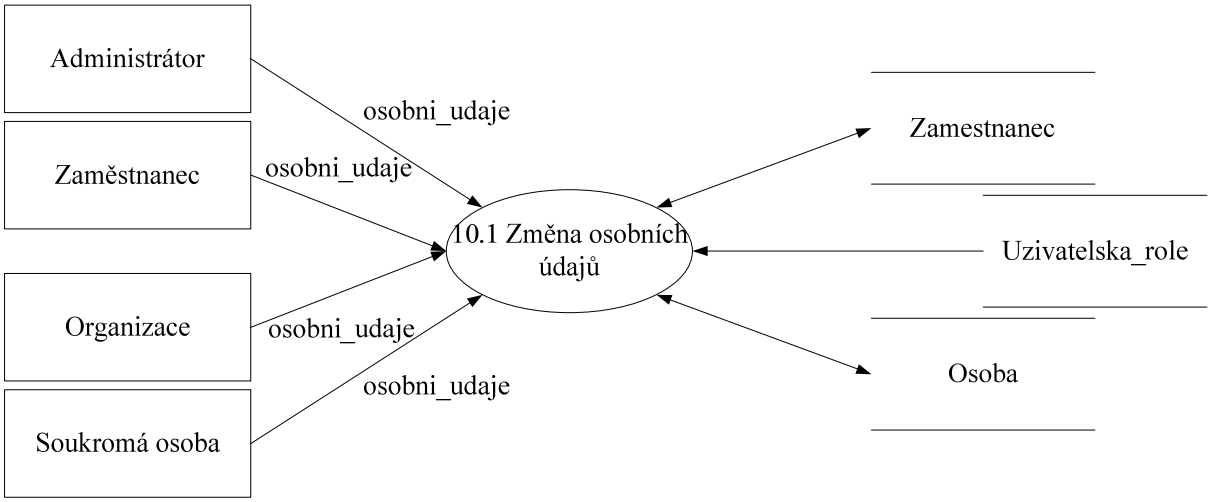
Obrázek 3.66: DFD (9.5 Výpis zaměstnance)

Algoritmus:

7. Zobraz formulář Přehled zaměstnanců (viz. 9.4 Výpis zaměstnanců)
8. Uživatel vybere zaměstnance, kterého chce detailně vypsat na obrazovku, kliknutím na tlačítko pro detail
9. Ulož id_zamestnanec vybraného zaměstnance do p.id_zamestnanec
10. Načti z tabulky Zamestnanec záznam, kde Zamestnanec.id_zamestnanec = p.id_zamestnanec a ulož hodnoty do p.id_role, p.titul_pred, p.jmeno, p.prijmeni, p.titul_za, p.ulice, p.cp, p.mesto, p.psc, p.telefon, p.email, p.zkratka, p.odstraneno
11. Doplň zaměstnance o atribut nazev_role z tabulky Uzivatelska_role, kde Uzivatelska_role.id_role = p.id_role

12. Vypiš záznam doplněný o výše uvedený atribut ve formuláři Zaměstnanec včetně odkazů pro další funkce spojené s vybraným zaměstnancem

10.1 Změna osobních údajů … funkce, která edituje osobní údaje přihlášeného zaměstnance (Administrátor, Zaměstnanec) nebo zákazníka (Organizace, Soukromá osoba).



Obrázek 3.67: DFD (10.1 Změna osobních údajů)

Algoritmus:

1. Ulož do p.id hodnotu ID aktuálně přihlášeného uživatele systému
2. Načti z tabulky Zamestnanec záznam, kde Zamestnanec.id_zamestnanec = p.id a ulož hodnoty do p.id_zamestnanec, p.id_role, p.titul_pred, p.jmeno, p.prijmeni, p.titul_za, p.ulice, p.cp, p.mesto, p.psc, p.telefon, p.email, p.zkratka, p.odstraneno
3. Jestliže neexistuje záznam v tabulce Zamestnanec, kde Zamestnanec.id_zamestnanec = p.id

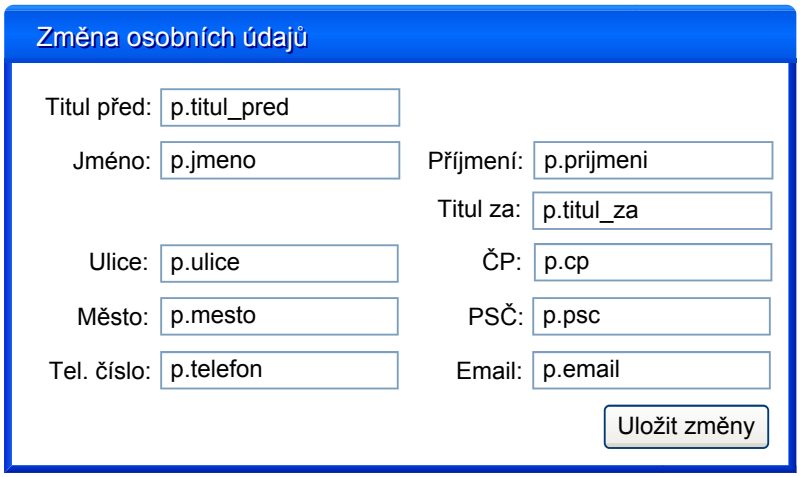
3.1. Načti z tabulky Osoba záznam, kde Osoba.id_osoba = p.id a ulož hodnoty do p.id_osoba, p.id_role, p.titul_pred, p.jmeno, p.prijmeni, p.titul_za, p.ulice, p.cp, p.mesto, p.psc, p.telefon, p.email

3.2. Jestliže neexistuje záznam v tabulce Osoba, kde Osoba.id_osoba = p.id

PAK

3.2.1. Vypiš chybovou hlášku „Nebyl nalezen záznam o přihlašeném uživateli“ a jdi na bod 1

4. Načti z tabulky Uzivatelska_role záznam, kde Uzivatelska_role.id_role = p.id_role a ulož do p.nazev_role, p.popis
5. Zobraz ve formuláři Změna osobních údajů načtené hodnoty z databáze



6. Uživatel edituje hodnoty ve formuláři Změna osobních údajů
7. Ulož hodnoty z formuláře Změna osobních údajů do p.titul_pred, p.jmeno, p.prijmeni, p.titul_za, p.ulice, p.cp, p.mesto, p.psc, p.telefon, p.email
8. Zkontroluj IO. Jestliže je chyba, vypiš chybovou hlášku a jdi na bod 5
9. Jestliže p.nazev_role = „Administrátor“ nebo p.nazev_role = „Zaměstnanec“

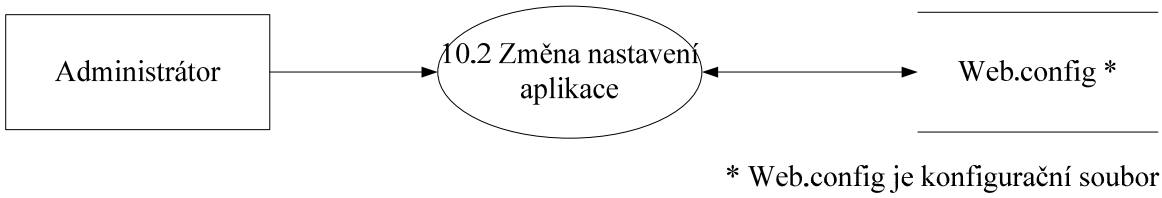
   PAK

   9.1. Proveď zápis údajů p.id_role, p.titul_pred, p.jmeno, p.prijmeni, p.titul_za, p.ulice, p.cp, p.mesto, p.psc, p.telefon, p.email, p.zkratka, p.odstraneno do tabulky Zamestnanec, kde Zamestnanec.id_zamestnanec = p.id_zamestnanec
10. Jestliže p.nazev_role = „Organizace“ nebo p.nazev_role = „Soukromá osoba“

    PAK

    10.1. Proveď zápis údajů p.id_role, p.titul_pred, p.jmeno, p.prijmeni, p.titul_za, p.ulice, p.cp, p.mesto, p.psc, p.telefon, p.email do tabulky Osoba, kde Osoba.id_osoba = p.id_osoba
11. Jestliže se zápis neprovedl, vypiš chybovou hlášku a jdi na bod 1

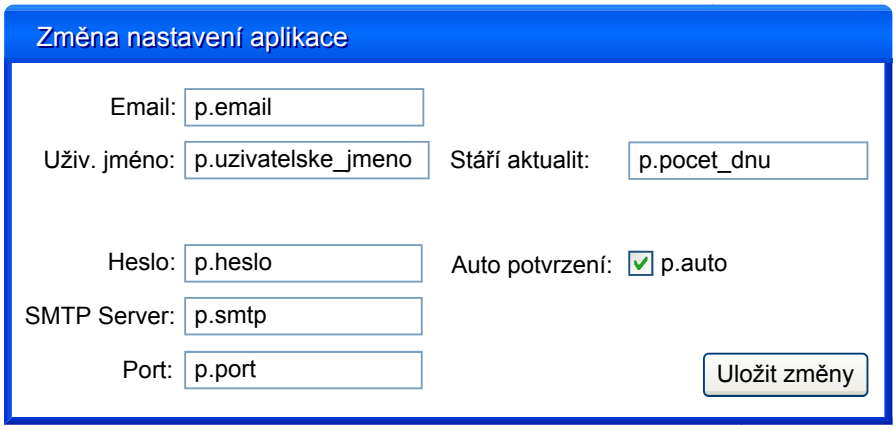
10.2 Změna nastavení aplikace … funkce, kterou může použít Administrátor pro změnu nastavení aplikace, tzn. pro změnu některých konfiguračních hodnot, které mají vliv na chování celé aplikace (např. nastavení emailové pošty aj.).



Obrázek 3.68: DFD (10.2 Změna nastavení aplikace)

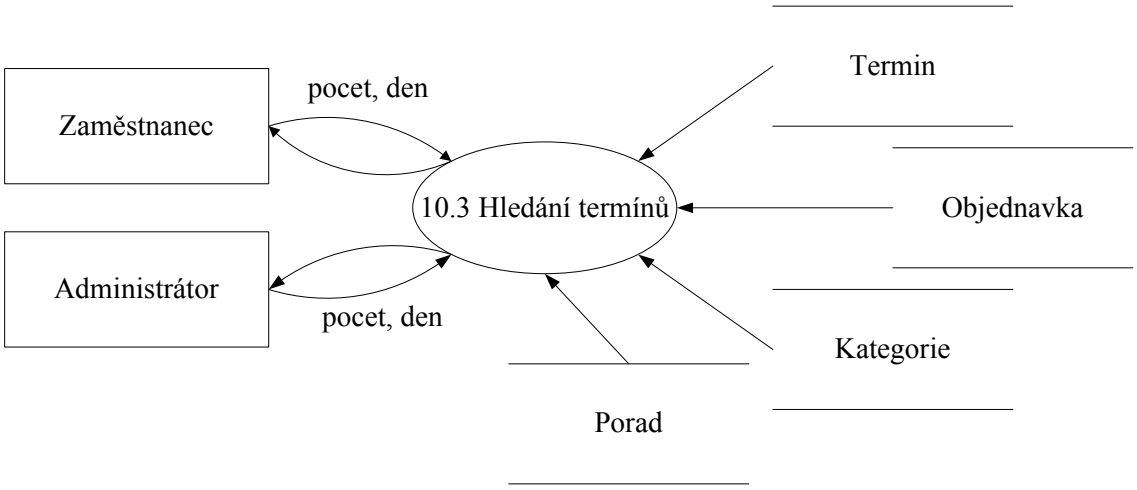
Algoritmus:

1. Načti z konfiguračního souboru Web.config paramerty aplikace, které mohou být editovány Administrátorem přes rozhraní aplikace a ulož do .p.email, p.uzivatelske_jmeno, p.heslo, p.smtp, p.port, p.pocet_dnu, p.auto
2. Zobraz formulář Změna nastavení aplikace



3. Uživatel modifikuje hodnoty ve formuláři Změna nastavení aplikace
4. Ulož hodnoty do p.email, p.uzivatelske_jmeno, p.heslo, p.smtp, p.port, p.pocet_dnu, p.auto
5. Zkontroluj IO. Jestliže je chyba, vypiš chybovou hlášku a jdi na bod 2
6. Proveď zápis údajů p.email, p.uzivatelske_jmeno, p.heslo, p.smtp, p.port, p.pocet_dnu, p.auto do konfiguračního souboru Web.config

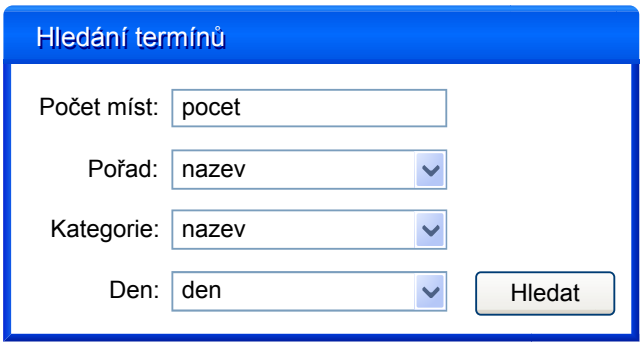
10.3 Hledání termínů … funkce, která hledá podle zadaných kritérií vhodný termín pro novou objednávku. Zadaná kritéria mohou být počet míst (viz. Objednavka), tj. myslí se počet volných míst na daný termín. Další kritéria jsou název dne v týdnu (viz. Termin), pořad a kategorie. Všechna kritéria jde mezi sebou kombinovat.



Obrázek 3.69: DFD (10.3 Hledání termínů)

Algoritmus:

1. Zobraz formulář Hledání termínů



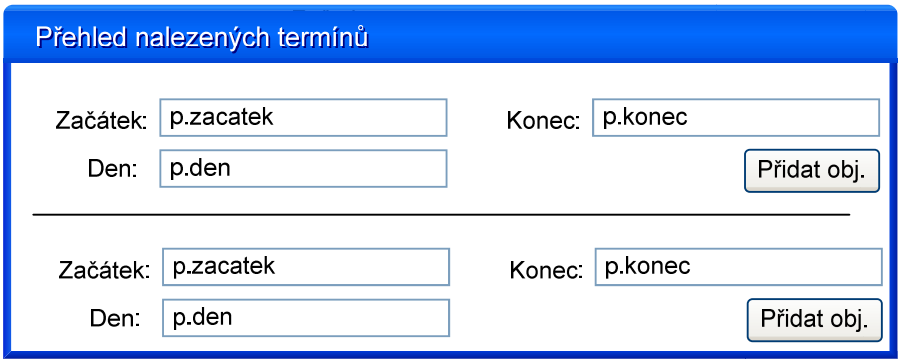
2. Uživatel vyplní hodnoty na formuláři Hledání termínů
3. Ulož hodnoty z formuláře Hledání termínů do p.pocet, p.den
4. Jestliže chce uživatel změnit kategorii
   PAK
   4.1. Zobraz seznam kategorií z tabulky Kategorie
   4.2. Uživatel vybere kategorii podle atributu nazev
   4.3. Ulož Kategorie.id_kategorie vybrané kategorie do p.id_kategorie
5. Jestliže chce uživatel změnit pořad
   PAK
   5.1. Zobraz seznam pořadů z tabulky Porad
   5.2. Uživatel vybere pořad podle atributu nazev
   5.3. Ulož Porad.id_porad vybreného pořadu do p.id_porad
6. Zkontroluj IO. Jestliže je chyba, vypiš chybovou hlášku a jdi na bod 1
7. Uživatel klikne na tlačítko Hledat
8. Načti z tabulky Termin všechny záznamy, kde Termin.den = p.den AND Termin.id_termin = Objednavka.id_termin AND Objednavka.id_porad = p.id_porad AND Objednavka.id_kategorie = p.id_kategorie AND SUM (Objednavka.pocet) GROUP BY

Objednavka.id_termin + p.pocet <= 100 AND Termin.uzaverka = FALSE a ulož do p.id_termin

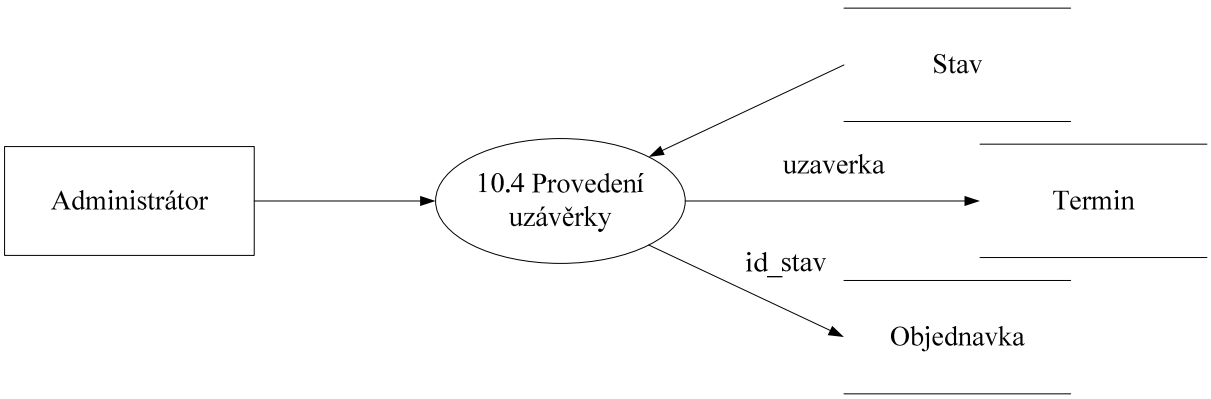
9. PRO KAŽDÉ p.id_termin DĚLEJ

9.1. Načti z tabulky Termin záznam, kde Termin.id_termin = p.id_termin a ulož do p.zacatek, p.konec, p.den, p.uzaverka, p.tisk

10. Zobraz seznam nalezených termínů ve formuláři Přehled nalezených termínů včetně odkazu pro přechod na funkci Přidání objednávky



10.4 Provedení uzávěrky … funkce, která se týká vždy jednoho týdne a znamená, že po provedení této funkce již nejsou možné další změny (objednávka, rezervace aj.) na tento vybraný týden. Tuto funkci provádí Administrátor a také pouze tato role má možnost udělat ještě další změny i po provedení uzávěrky (např. z důvodu nějaké opravy údajů apod.).



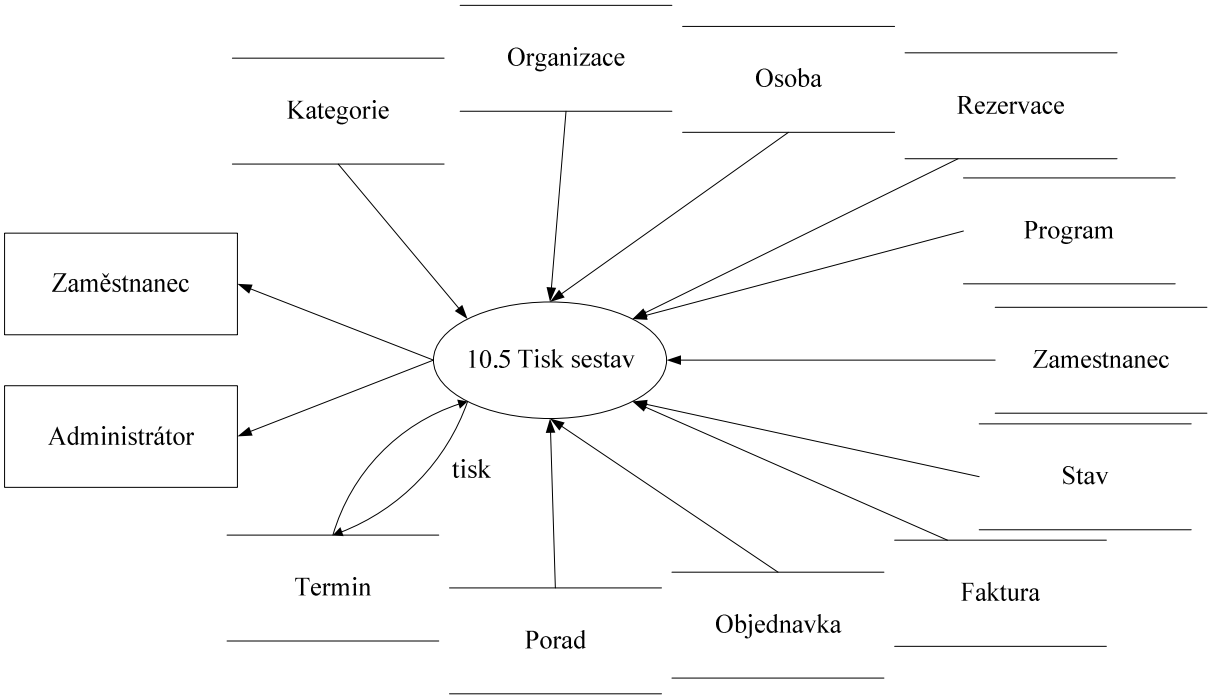
Obrázek 3.70: DFD (10.4 Provedení uzávěrky)

Algoritmus:

1. Proveď funkci Přehled objednávek (viz. 2.4 Přehled objednávek) nebo funkci Přehled programů pro veřejnost (viz. 7.4 Přehled programů pro veřejnost)
2. Z hodnoty p.datum, která je zadaná v záhlaví formuláře, zjisti počátek týdne a konec týdne a ulož do p.zacatek a p.konec
3. Uživatel klikne na tlačítko Uzávěrka
4. Nastav do p.uzaverka hodnotu TRUE
5. Načti z tabulky Stav záznam, kde Stav.nazev_stavu = „Potvrzená“ a ulož hodnoty do p.id_stav, p.nazev_stavu
6. BEGIN TRANSACTION

7. Proveď zápis údaje p.uzaverka do tabulky Termin, kde Termin.zacatek >= p.zacatek AND Termin.konec <= p.konec
8. Proveď zápis údaje p.id_stav do tabulky Objednavka, kde Objednavka.id_termin = Termin.id_termin AND Termin.zacatek >= p.zacatek AND Termin.konec <= p.konec AND Objednavka.id_stav = Stav.id_stav AND (Stav.nazev_stavu = „Nová” OR Stav.nazev_stavu = „Upravená”)
9. END TRANSACTION
10. Jestliže se zápis neprovedl, vypiš chybovou hlášku a jdi na bod 1

10.5 Tisk sestav … obecná funkce, která sestaví tiskovou sestavu podle zadaného typu sestavy. Uživatel klikne na tlačítko pro tisk sestavy a na obrazovku se zobrazí tisková sestava, podle typu tlačítka, na které kliknul.



Obrázek 3.71: DFD (10.5 Tisk sestav)

Algoritmus:

1. Proveď funkci Přehled objednávek (viz. 2.4 Přehled objednávek) nebo funkci Přehled programů pro veřejnost (viz. 7.4 Přehled programů pro veřejnost)
2. Z hodnoty p.datum, která je zadaná v záhlaví formuláře, zjisti počátek týdne a konec týdne a ulož do p.zacatek a p.konec
3. Uživatel klikne na jedno z tlačítek pro tisk výstupní sestavy
4. Nastav do p.tisk hodnotu TRUE
5. Proveď zápis údaje p.tisk do tabulky Termin, kde Termin.zacatek >= p.zacatek AND Termin.konec <= p.konec
6. Načti z tabulky Objednavka nebo z tabulky Program všechny záznamy, kde (Objednavka.id_termin = Termin.id_termin AND Termin.zacatek >= p.zacatek AND

Termin.konec <= p.konec) OR (Program.id_termin = Termin.id_termin AND Termin.zacatek >= p.zacatek AND Termin.konec <= p.konec) a doplň záznamy o atributy z dalších tabulek podle cizích klíčů Objednavka.id_osoba, Objednavka.id_organizace, Objednavka.id_faktura, Objednavka.id_termin, Objednavka.id_porad, Objednavka.id_kategorie, Objednavka.id_zamestnanec, Objednavka.id_stav, resp. Program.id_termin, Program.id_porad, Program.id_zamestnanec

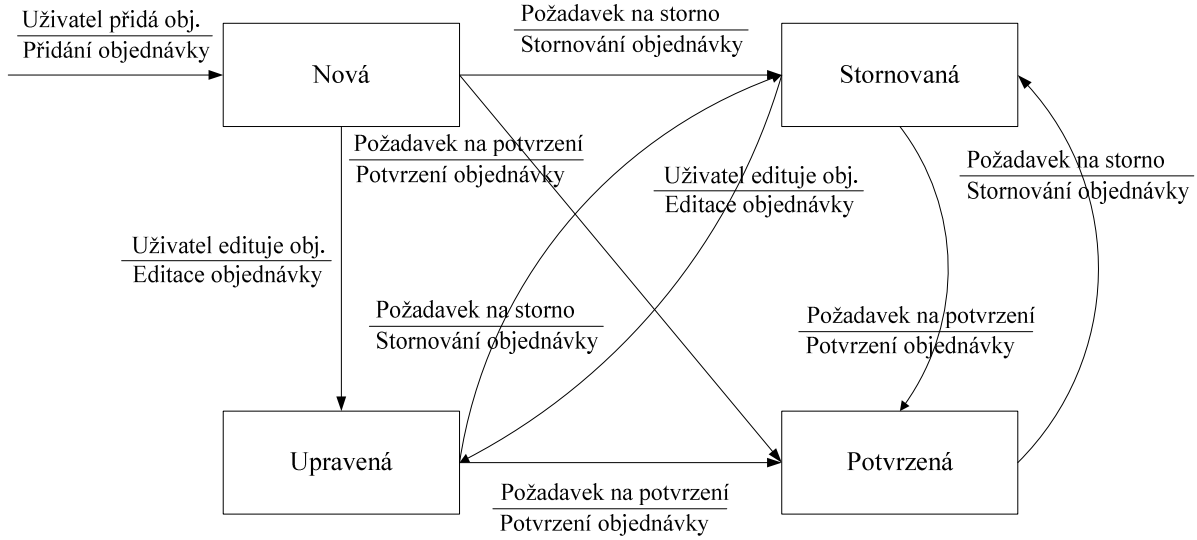
7. Načti z tabulky Rezervace všechny záznamy, kde Rezervace.id_program = Program.id_program načtený v předchozím dotazu a doplň záznamy o atributy z dalších tabulek podle cizích klíčů Rezervace.id_osoba, Rezervace.id_organizace
8. Zobraz všechny záznamy načtené v předcházejících dotazech ve formuláři pro tisk sestavy, podle zvoleného typu sestavy (viz. Přílohy A)

3.2.3 Datové toky

Kapitola datové toky obsahuje datový slovník (viz. 3.1.3) doplněný o datové toky použité v předcházejících DFD diagramech elementárních funkcí systému pro HaPJP. Datové toky můžete najít v přílohách této diplomové práce (viz. Přílohy C).

3.3 Dynamická analýza

Dynamická analýza popisuje časové návaznosti jednotlivých funkcí (procesů) a stavy jednotlivých entit nebo případně celého systému. Obsahem dynamické analýzy bude stavový diagram STD (viz. Obrázek 3.72) znázorňující stavy a přechody mezi stavy pro libovolnou objednávku evidovanou realizovaným IS. Jednotlivé stavy budou pro upřesnění ještě slovně popsány.



Obrázek 3.72: Stavový diagram pro objednávku

Stavy entity Objednávka:

- Nová
- Upravená
- Potvrzená
- Stornovaná

Každá objednávka se nachází, po přidání do databáze IS, ve stavu Nová. Tento stav indikuje, že objednávka byla do systému přidána a nebyly s ní provedeny žádné další akce. Ze stavu Nová se může objednávka dostat do všech ostatních stavů. Pokud uživatel modifikuje záznam o objednávce, je automaticky nastaven stav objednávky na Upravená. Tento stav indikuje nějaké změny v objednávce v porovnání s novou objednávkou. Jelikož je systém zamýšlen jako webový, tj. dostupný online, a registrovaný uživatel (Organizace, Soukromá osoba) může vytvořit objednávku nebo ji editovat, tak je důležitá indikace stavu objednávky. Při každé změně (přidání, editace) je generována emailová zpráva o této události, která je zaslána na emailovou adresu zaměstnance HaPJP. Tato emailová adresa je součástí nastavení systému. Ze stavu Nová nebo Upravená se může objednávka dostat do stavů Potvrzená nebo Stornovaná. Tyto přechody jsou způsobeny uživatelským rozhodnutím a následným kliknutím na tlačítko pro potvrzení, resp. stornování objednávky. Objednávka se stane potvrzenou také při provedení funkce Uzávěrka, která způsobí převod všech objednávek v daném týdnu do stavu Potvrzená, pokud ovšem není objednávka již ve stavu Stornovaná. Pokud ano, neprovede se tento přechod. Objednávka může být zpětně ze stavu Stornovaná převedena do stavu Potvrzená. Podmínkou je uživatelská interakce. Všechny stavy mají za úkol informovat uživatele o tom, v jakém stavu se

objednávka právě nachází. Pokud je objednávka ve stavu Stornovaná, pak není zobrazena v přehledech, tiskových sestavách a není zahrnuta do výsledných statistik.

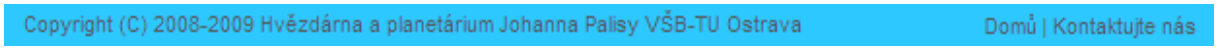
3.4 Návrh ovládání a komunikace s uživatelem

Návrh ovládání a komunikace s uživatelem je kapitola, která obsahuje návrh způsobu práce s novým informačním systémem. Kapitola také popisuje nejrůznější části a detaily uživatelského vzhledu programu, které se týkají ovládání a designu celé aplikace. Některé části budou prezentovány graficky, jiné budou popsány pouze slovně.

Záhlaví programu (viz. Obrázek 3.73) obsahuje logo HaPJP a také tlačítka pro obecné funkce systému. Barevné provedení záhlaví odpovídá barvám specifikovaným v grafickém manuálu (viz. Obsah přiloženého CD). Jelikož bude systém řešený jako webový informační systém, tak logo v záhlaví bude fungovat jako odkaz na domovskou webovou stránku celé aplikace. Další tlačítka slouží pro přechod na stránku s přihlašovacím dialogem, pro otevření nápovědy a pro přechod na domovskou stránku celé organizace.

Obrázek 3.73: Návrh vzhledu programu (záhlaví systému)

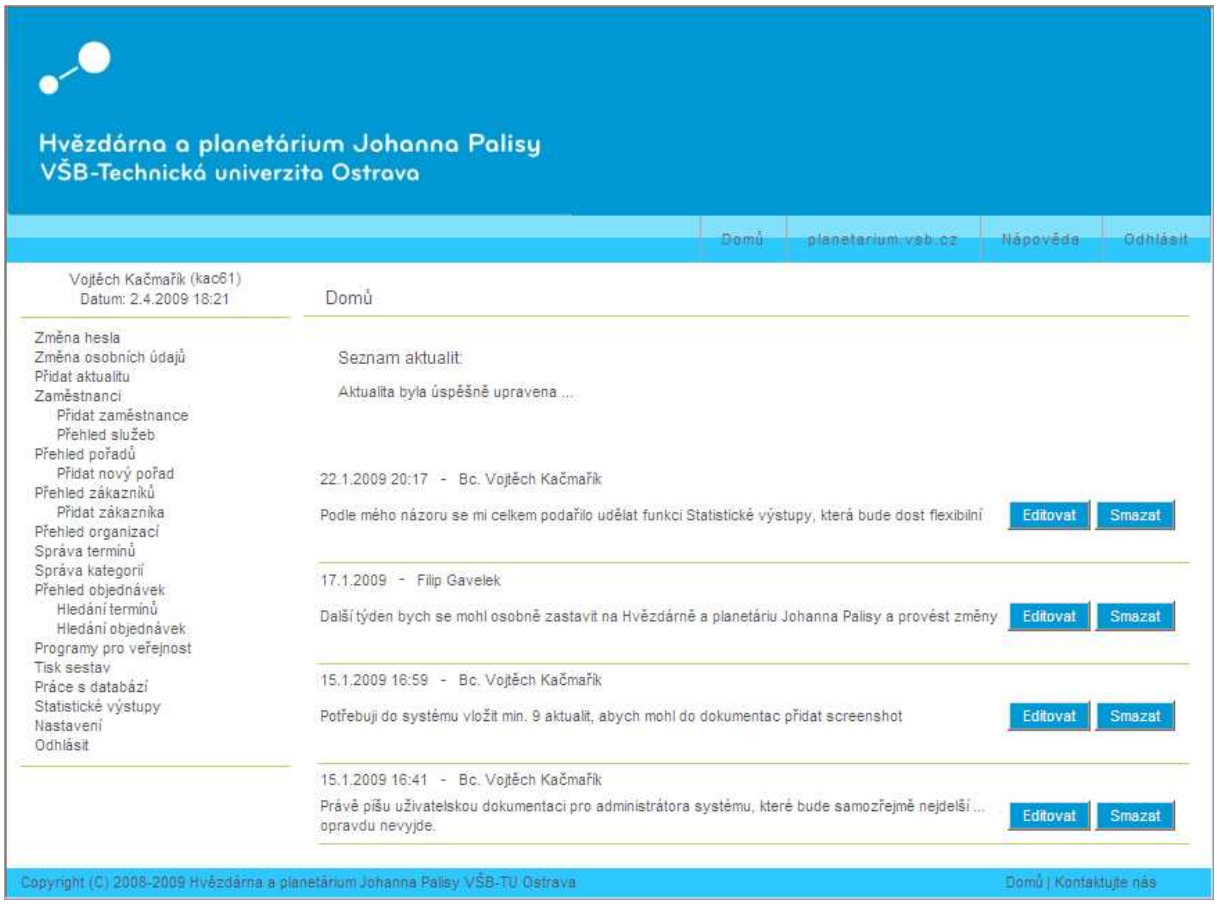
Zápatí programu (viz. Obrázek 3.74) obsahuje pouze malou lištu s informací o systému a odkazy pro přechod na domovskou stránku a pro zaslání emailové zprávy na adresu HaPJP.



Obrázek 3.74: Návrh vzhledu programu (zápatí systému)

Mezi záhlavím a zápatím bude hlavní použitelná plocha celého IS. V této části bude umístěno hlavní uživatelské menu systému a hlavní prostor pro vstupní a výstupní formuláře systému podle množiny minispecifikací, uvedených v kapitole Funkční analýza (viz. 3.2). Uživatelské menu bude uvedeno v levé části obrazovky a bude poskytovat odkazy na jednotlivé funkce, resp. webové stránky, aplikace (viz. Obrázek 3.75). Uživatelské menu se bude generovat dynamicky na základě uživatelské role přihlášeného uživatele. Hlavní část systému bude potom soustředěna do zbytku plochy na obrazovce. Pod záhlavím bude zobrazena cesta včetně odkazů, kde se uživatel zrovna nachází. Dále je na obrázku vidět i řešení výstupních informací na obrazovku. V záhlaví každé obrazovky bude název funkce, nebo výstupu. Pod tímto údajem je volné místo, kde se bude uživateli zobrazovat informativní nebo chybová zpráva podle průběhu vybrané funkce. Chybová zpráva bude zobrazena červeně. Výstupní informace budou zobrazeny prostým textem o velikosti 11px a standartním bezpatkovým písmem Arial, Helvetica nebo sans-serif. Celá aplikace se bude ovládat pomocí myši i klávesnice. Při

zadávání vstupních informací se použije vstup z klávesnice. Na přechod mezi jednotlivými vstupními poli může být použit tabelátor. Pro potvrzení funkce, nebo pro zahájení další funkce apod. jsou použita tlačítka. Uživatel pomocí těchto tlačítek intuitivně ovládá další chování celého IS. Všechna tlačítka mají stejný vzhled v celé aplikaci. Nápověda pro vstupní pole je řešena formou plovoucí nápovědy, tzv. tooltip. Pokud uživatel podrží kurzor myši nad vstupním polem, tak je mu následně zobrazena nápověda k významu tohoto pole.



Obrázek 3.75: Návrh vzhledu programu (hlavní obrazovka)

Pokud bude uživatel v systému provádět odstranění nějakého záznamu, pak mu bude nabídnuto standartní okno pro odstranění záznamu (viz. Obrázek 3.76). Uživatel musí tuto akci potvrdit, aby nedošlo k nechtěnému odstranění záznamu z databáze IS.

Obrázek 3.76: Návrh vzhledu programu (standartní message box)

4. Návrh implementace

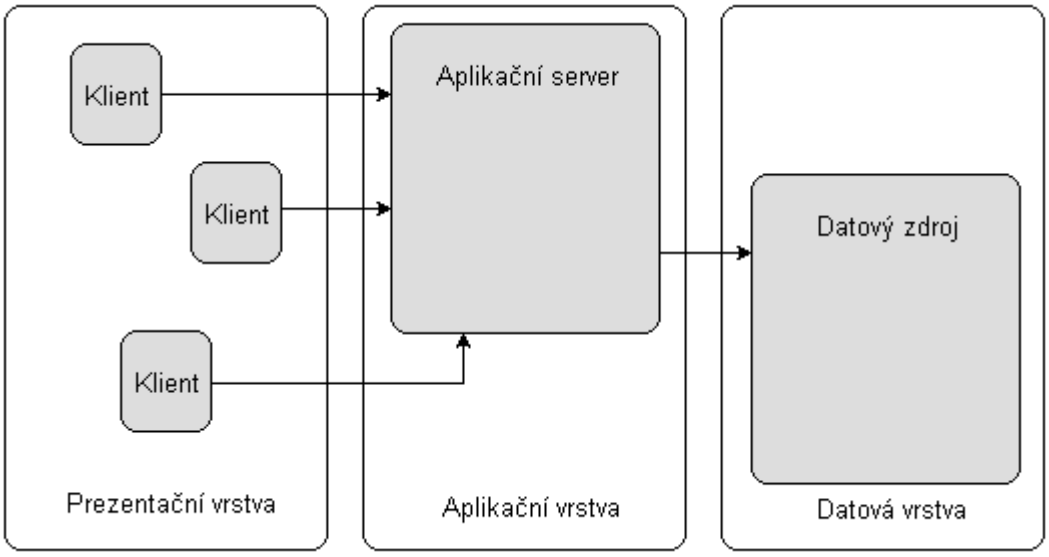
Kapitola Návrh implementace bude popisovat, jak se bude celý analyzovaný systém skutečně realizovat. Řeší se jednak další detaily v již zpracovaných modelech (datový, funkční), aby implementace měla optimální vlastnosti, jednak se nyní berou v úvahu dosud nepoužité nefunkční požadavky ze specifikace požadavků (viz. [Šarm 4]). Výstupem návrhu implementace bude detailní zadání pro rutinní implementaci. Návrh implementace bude rozdělen na 2 úrovně, tj. systémový návrh, který řeší např. návrh HW a SW prostředí, a vlastní návrh implementace, tj. upřesnění, doplnění a optimalizace algoritmů, doplnění dat, rozdělení funkcí do modulů aj.

4.1 Systémový návrh

Systémový návrh jako úvodní koncepční část návrhu implementace vychází z výsledků analýzy a z nefunkčních požadavků. Z nefunkčních požadavků se zde berou v úvahu především požadavky na použité HW a SW prostředí. Tyto informace nebyly dříve detailně specifikovány, pouze bylo stanoveno, že z důvodu on-line dostupnosti bude nová aplikace řešena jako webový informační systém a náklady na provozní prostředí by měly být co nejmenší.

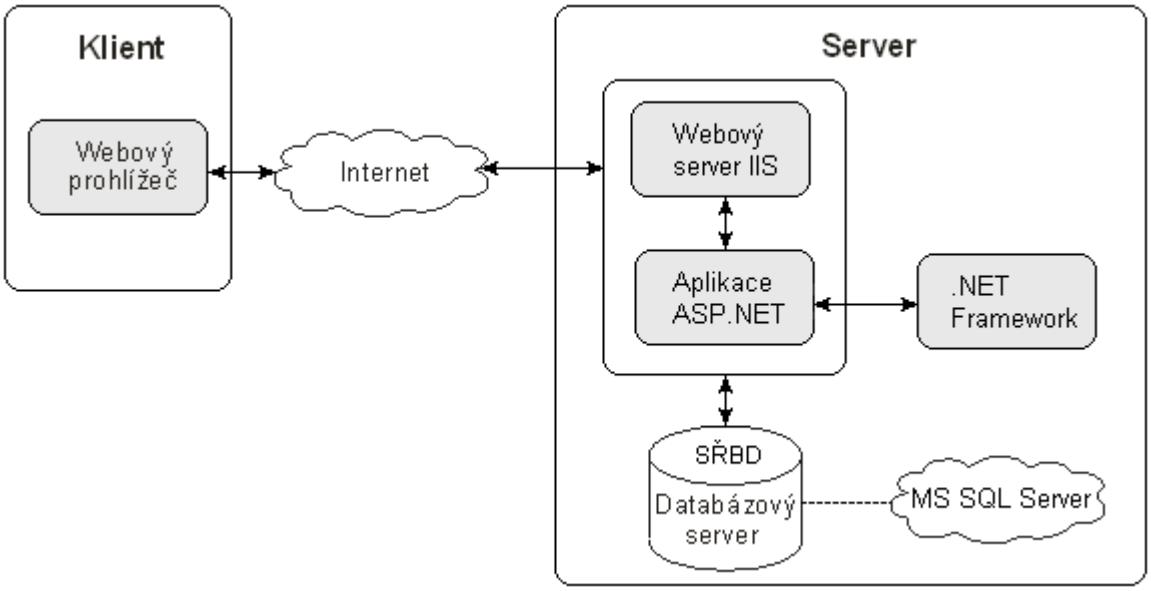
4.1.1 Architektura informačního systému

Jelikož bylo stanoveno, že implementovaná aplikace bude webovým informačním systémem, bude zde prezentována architektura webové aplikace. Jedná se o typickou třívrstvou architekturu skládající se z webového prohlížeče (prezentační vrstva, presentation layer), webového serveru, na kterém je umístěna aplikační logika (aplikační vrstva, business layer), a databázového serveru (databázová vrstva, data layer). Obecný princip třívrstvé architektury je vidět na následujícím obrázku (viz. Obrázek 4.1). Klienti nemají žádné informace o logice aplikace, která je umístěna na aplikačním serveru a s pomocí datové vrstvy obstarává požadavky klientů.



Obrázek 4.1: Obecný princip třívrstvé architektury

Tento obecný princip třívrstvé architektury bude využit u architektury naší webové aplikace. Princip webové aplikace HaPJP je následující. Klienti aplikace budou mít na svém počítači webový prohlížeč (např. Mozilla, Internet Explorer), který pomocí HTTP protokolu komunikuje přes síťové rozhraní s webovým serverem. Webovým serverem je v našem případě myšlen fyzický stroj s operačním systémem Microsoft Windows Server 2003, nebo Microsoft Windows XP. Počítač je připojen do sítě a běží na něm tzv. virtuální webový server, tj. Microsoft IIS (Internet Information Services). Jestliže požadavek klienta směřuje na statickou stránku, pak webový server vyhledá daný soubor a vrátí odpověď klientovi. Pokud si klient vyžádá dynamickou stránku, předá webový server požadavek aplikačnímu serveru (ASP.NET), který požadavek přijme a stará se o jeho vyřízení. Při dynamickém zpracovávání požadavku může aplikační server komunikovat s databázovým serverem (MS SQL Server 2005), odkud získá potřebná data. Výsledkem zpracování požadavku je odpověď v podobě (X)HTML stránky, která je předána webovému serveru a ten ji pošle zpět klientovi pomocí protokolu HTTP. Přijatá stránka je pak zobrazena ve webovém prohlížeči klienta (viz. Obrázek 4.2). Pro dynamické zpracování požadavku pomocí ASP.NET musí být na fyzickém stroji nainstalován také .NET Framework.



Obrázek 4.2: Princip webové aplikace

4.1.2 .NET Framework

.NET Framework je nová platforma pro budování systémů na rodině operačních systémů Windows, ale také na četných operačních systémech nepocházejících od společnosti Microsoft, jako Mac OS X a různé distribuce Unix/Linux. Z hlediska programátora se dá .NET chápat jako runtime prostředí a vyčerpávající knihovna základních tříd (viz. [C# 2]). Společnost Microsoft vyvinula specificky pro tuto platformu nový programovací jazyk nazvaný jako C#. Programovací jazyk C# má podobnou syntaxi jako Java (ale není s ní identický). Platforma .NET podporuje také jiné programovací jazyky, jako např. Visual Basic .NET, J# aj.). Bez ohledu na zvolený programovací jazyk je potom zdrojový kód odpovídajícím kompilátorem .NET převeden na tzv. assembly, což je

binární jednotka .NET (.dll nebo .exe), která v sobě obsahuje tzv. intermediární jazyk CIL (někdy označován také jako IL) a metadata. Tento jazyk je zcela nezávislý na platformě. Při samotném použití této nezávislé assembly je potom CIL převeden kompilátorem JIT (just-in-time, „právě včas“) do smysluplných instrukcí CPU na použitém běhovém prostředí.

4.1.3 ASP.NET

Technologie ASP.NET je nedílnou součástí .NET Frameworku, která slouží k vytváření webových aplikací běžících na serveru. Nejedná se tedy o programovací jazyk, ale o soubor funkcí a objektů, které umožňují vytvářet webové stránky pracující na serveru (viz. [ASP.NET 2]). Je to tedy plně objektově orientovaná technologie. Vlastní stránky se pak programují v některém z podporovaných programovacích jazyků (obvykle C# nebo Visual Basic .NET). ASP.NET není, na rozdíl od většiny skriptovacích jazyků, interpretovaný, ale je kompilovaný pomocí CLR (Common Language Runtime) do strojového kódu počítače.

ASP.NET je založen na modelu klient/server, ale současně využívá i rysy událostmi řízeného programování, které se používá převážně v klasických aplikacích Windows. Princip ASP.NET spočívá právě v těchto událostech. Pokud se na stránce v prohlížeči objeví nějaká událost, např. uživatel stiskne tlačítko nebo vyplní nějakou hodnotu apod., odešle prohlížeč informace o této události na server, kde je možné na ni zareagovat, například změnou obsahu stránky. Takto změněná stránka se odešle zpět do prohlížeče. Tento postup zaručuje, že stránky vytvořené v ASP.NET jsou maximálně interaktivní a mohou se dynamicky měnit v závislosti na akcích uživatele.

4.1.4 IIS (Internet Information Services)

Webovým serverem, který se používá ve většině scénářů spolu s ASP.NET, bývá ten, který se dodává s Windows – IIS. Pomocí IIS můžeme nakonfigurovat, které adresáře aplikace budou vystaveny jako virtuální adresáře, a budou tedy přístupné ostatním klientům, kteří je budou požadovat přes síť nebo přes Internet. IIS je v podstatě služba Windows. Má na starost zpracování požadavků, které obdrží na konkrétních portech. K tomuto účelu běží v systému služba nazvaná World Wide Web Publishing Service (Služba publikování na WWW), která naslouchá na několika síťových portech TCP/IP (obvykle port 80 HTTP a port 443 HTTPS). Tuto službu spravuje Konzola IIS (IIS management console), kde můžeme vytvářet a konfigurovat různé weby (viz. [ASP.NET 3]). ISS je automaticky instalováno na operační systém Windows Server a je volitelnou součástí Windows XP. Oba tyto operační systémy nám budou stačit pro běh naší aplikace, nicméně doporučováno je mít Windows Server.

4.1.5 Microsoft SQL Server 2005

Pro uchovávání dat potřebujeme mít databázi. Samotná databáze by bez programu, který s databází spolupracuje, byla k ničemu. Proto budeme dále mluvit pouze o SŘBD (systém řízení báze dat), což je programový systém umožňující definovat datové soubory, manipulovat s daty, formátovat vstupní a výstupní informace atd. Takový systém je zcela nezávislý na datech a řídí přístup právě k těmto datům. Pomocí SŘBD lze zajistit současnou práci více uživatelů apod. Při výběru optimálního SŘBD máme spoustu možností, který zvolit. Mezi nejznámější komerční SŘBD se řadí Microsoft

SQL Server, Oracle, DB2, Sybase aj. Pro naši aplikaci jsem vzhledem k použitým technologiím vybral MS SQL Server 2005 (viz. [MS SQL 2005]) v jeho Express edici, která je zdarma a pro naše účely bohatě postačí.

4.2 Vlastní návrh implementace

Součástí vlastního návrhu implementace, nebo taky objektového návrhu, bude úprava a doplnění funkcí z funkční analýzy (např. transakční analýza, indexová analýza), zabezpečení mezního provozu (např. zálohování dat, bezpečnost, výkon) a doplnění systémových funkcí (např. logování aj.)

4.2.1 Upřesnění algoritmů minispecifikací

Minispecifikace uvedené v kapitole Funkční analýza (viz. 3.2) neobsahují všechny detaily, které je třeba brát v potaz při následné implementaci. Podle použitého SŘBD, tj. MS SQL Server 2005, je stanoveno, že primární klíče ID jsou téměř u všech tabulek tzv. IDENTITY, což je obdoba autoincrement z jiných SŘBD. Jedná se o to, že tato hodnota je automaticky inkrementována při vložení nového záznamu do databáze. Proto nejsou hodnoty ID v minispecifikacích uvažovány. Pouze tabulky Osoba a Zamestnanec obsahují ID, které se nevytvářejí automaticky. Proto je postup práce s primárním klíčem těchto tabulek popsán v minispecifikacích.

Jednotlivé minispecifikace také neobsahují přesný popis IO, které jsou v průběhu jednotlivých funkcí kontrolovány. Popis IO je uveden v datovém slovníku u jednotlivých atributů každé tabulky, pokud je nějaké omezení definováno.

4.2.2 Indexová analýza

Na základě provedené analýzy minispecifikací byly vytvořeny databázové indexy a potřebné informace o těchto indexech byly doplněny do datového slovníku (viz. 3.1.3). Indexované atributy se používají pro vyhledávání nebo pro třídění a existující indexy mají vliv na rychlost provedení dotazu do databáze. Všechny uvedené indexy jsou udržované. V systému není žádná funkce, která by se používala opakovaně (z hlediska časové periody, měsíčně, čtvrtletně apod.) a bylo by tedy výhodné zavést index dočasný. Minispecifikace tedy nebyly doplněny o žádné příkazy CREATE INDEX nebo DROP INDEX.

Pro vytvoření indexu v MS SQL Server 2005 můžeme použít syntaxi, která je uvedena níže. V příkladu, který následuje, jsem záměrně uvedl vytvoření indexu skládajícího se ze dvou atributů. Tento index je primárně setříděn podle atributu [jmeno] a sekundárně podle atributu [prijmeni], které se oba nacházejí v tabulce Zamestnanec a je pojmenován jako [Zamestnanec_Jmeno].

```
CREATE INDEX [index_name] ON [table_name] ([attr1_name], [[attr2_name], [attr3_name], …]])

CREATE INDEX [Zamestnanec_Jmeno] ON [Zamestnanec] ([jmeno] , [prijmeni])
```

4.2.3 Analýza zálohování databáze

Tato kapitola se bude věnovat zálohování databáze, resp. dat obsažených v databázových tabulkách. Je třeba stanovit způsob zálohování a také periodu, což je u tohoto IS velmi důležité, neboť

na tento požadavek byl kladen zadavatelem velký důraz. Zadavatel se obává ztráty dat, a proto je pro něj záloha velmi důležitá.

Jelikož je aplikace webovým IS, je tedy dostupná nepřetržitě. Pro zálohování bychom měli stanovit čas, kdy je systém co nejméně zatížen a samotné zálohování tak bude mít co nejmenší dopad na výkon systému. Jelikož zadavatel požaduje provádění zálohy několikrát denně, stanovil jsem časy provedení zálohy na 7:30, 15:30 a 23:30. Zálohovat se bude vždy celá databáze, tj. struktura databáze včetně obsahu jednotlivých tabulek. Pro provedení zálohy je vytvořen spustitelný soubor backup.bat, který je s pomocí plánovače úloh v operačním systému Windows spouštěn denně v pravidelných intervalech. Záloha se ukládá do adresáře C:\HaPJP\Backup\ a vytvořený soubor zálohy obsahuje v názvu datum a čas vytvoření. Administrátor, který je zodpovědný za běhové prostředí systému (provozní server), může dále s těmito zálohami nakládat. Např. mohou být automaticky kopírovány na nějaké diskové pole, CD/DVD média apod. Zálohu systému může provést také samotný uživatel IS HaPJP, který je v roli Administrátora. Pro tuto funkci je v systému vytvořeno rozhraní, s pomocí kterého se záloha provede.

Samotné webové rozhraní poskytuje také funkci pro obnovení databáze z již provedené zálohy. Po provedení funkce je vytvořen v adresáři C:\HaPJP\Restore\ spustitelný soubor restore.bat, který obsahuje potřebné příkazy pro provedení samotné obnovy databáze. Administrátor serveru spustí tento soubor a automaticky se provede obnovení databáze z nejnovější zálohy systému. Administrátor serveru můžu také databázi zálohovat nebo obnovit přímo pomocí použitého SŘBD, v našem případě MS SQL Server 2005 Express. Slouží k tomu aplikační rozhraní aplikace MS SQL Server Management Studio Express.

4.2.4 Evidence chyb / Evidence využívání funkcí IS

Pro potřeby dohledávání chyb aplikace nebo uživatelských chyb při práci s novým systémem jsem navrhl logování chyb a uživatelských akcí do textových souborů. Při každé chybě, která je v systému evidována, se zapíše nová zpráva do logovacího souboru. Soubory jsou ukládány do adresáře C:\HaPJP\Log\ a název souboru je vždy HaPJPyyyyMMdd.log (např. HaPJP20090315.log). Uchovávají se informace o datu, času, typu akce, přihlášeném uživateli a detailním popisu akce, resp. chyby. Stejné informace se ukládají také při logování uživatelských akcí, tj. využívání funkcí IS. Z toho se dá potom analyzovat využívání funkcí systému apod.

Pro potřeby logování do textového souboru jsem navrhl, aby se při implementaci použil návrhový vzor Singleton. Singleton se v knize Návrh program pomocí vzorů (viz. [GoF]) řadí k takzvaným tvořivým objektovým vzorům, z čehož vyplývá, že je určen pro tvorbu objektu. Tento dessign pattern lze aplikovat všude tam, kde je požadován (nebo je vhodný) výskyt jediné instance třídy v rámci celé aplikace a kde tato instance má být klientům snadno dostupná. Detaily o základní implementaci tohoto vzoru v .NET najdete na MSDN Library (viz. [Singleton]).

4.2.5 Role uživatelů a přístupová práva

Podle uživatelské role musíme v systému rozlišovat množinu funkcí, které jsou pro danou roli dostupné a mohou být uživatelem spuštěny. Na základě této role bude uživateli dynamicky

vygenerováno menu obsahující pouze odpovídající funkce. V systému potřebujeme evidovat jednotlivé uživatelské role a jejich práva, resp. funkce, které jsou těmto rolím přiděleny. Dále musíme evidovat jednotlivé uživatele včetně loginů a hesel, a jejich přiřazení k určitým uživatelským rolím.

Pro tento účel existuje tzv. API pro členství (Membership API) ASP.NET, které poskytuje kompletní sadu funkcí pro správu uživatelů, kterou lze snadno a rychle integrovat (viz. [ASP.NET 3]). S využitím formulářové autentizace pro bezpečné přihlašování uživatelů do aplikace tak API pro členství poskytuje snadný způsob implementace správy uživatelů a s tím související funkcionality do naší aplikace. API pro členství pracuje zcela nezávisle na svém podkladovém úložišti dat. Úložiště dat může být buď přímo v naší navržené databázi, nebo v jiné databázi na našem provozním MS SQL Serveru. Pro používání této techniky se v úložišti dat vytvoří několik nových databázových tabulek a několik systémových funkcí, které jsou následně využívány programově skrz třídu Membership.

API pro členství se používá pouze pro správu uživatelů a ověřování jejich totožnosti. Neimplementuje funkcionalitu pro autorizaci a neposkytuje funkcionalitu pro správu uživatelských rolí. Pro tento účel musíme v ASP.NET použít API pro role (Roles API). API pro role obsahuje integrované funkce pro správu rolí, přidělování rolí uživatelům a pro přístup ke všem informacím o rolích z kódu. Třída Roles potom poskytuje kompletní programátorský přístup k informacím o rolích v aplikaci (viz. [ASP.NET 3]). API pro role se v ASP.NET používá výhradně ve spojení s API pro členství.

4.2.6 Transakční analýza

Tato kapitola týkající se návrhu transakcí má bezprostřední vliv na minispecifikace elementárních funkcí systému. Po provedení transakční analýzy byly doplněny do minispecifikací příkazy BEGIN TRANSACTION a END TRANSACTION. Obecně se transakce týkají vždy pouze změn v databázi a jejich úkolem je udržet data v neustále platném a konzistentním stavu.

Samotná implementace transakcí podle uvedených minispecifikací může být provedena buď na úrovni použitého SŘBD (v našem případě MS SQL Server 2005), nebo přímo v programovacím jazyce (např. C#, VB .NET). Z hlediska výkonu aplikace ASP.NET by se transakce měly psát na úrovni SQL, v našem případě by to bylo na úrovni uložených procedur v MS SQL Server 2005. Navíc platí, že používání transakcí, které nejsou nutné, může uškodit výkonu a použitelnosti aplikace. Proto by měly být použity opravdu pouze ty transakce, které jsou nezbytné z hlediska zachování konzistence dat.

Transakce také souvisí s tzv. simultánním (souběžným) zpracováním dat. Jde o zpracování záznamu více uživateli najednou, což se týká pouze víceuživatelských aplikací (to je náš případ). Při tomto způsobu zpracování může docházet k různým problémům v databázi, nekonzistenci apod. Proto musíme určit strategii simultánního zpracování (concurrency strategy). Pro správu simultánního zpracování existuje několik dobře známých přístupů (viz. [ASP.NET 3]). Jde o to, abychom si uvědomili, že tuto strategii určujeme tím, jak napíšeme příkaz UPDATE, resp. klauzuli WHERE. Já jsem pro naši víceuživatelskou aplikaci zvolil aktualizace záznamů založené na časových známkách (timestamps). MS SQL Server 2005 podporuje, stejně jako většina dnešních SŘBD, sloupce časových známek, které jsou zdrojem dat automaticky aktualizovány vždy, když se provede nějaká změna. Pro naši aplikaci to znamená, že každá tabulka z datového slovníku (viz. 3.1.3) bude obsahovat navíc nový

atribut [posledni_verze] datového typu TIMESTAMP. Klauzule WHERE potom bude obsahovat primární klíč aktualizovaného záznamu a aktuální časovou známku. Tím zajistíme, že záznam bude aktualizován pouze tehdy, pokud nebyl modifikován jiným (souběžným) uživatelem.

4.2.7 Bezpečnost systému

Dalším důležitým aspektem návrhu aplikace je její bezpečnost. ASP.NET již má v sobě zabudovanou základní strukturu poskytující bezpečnostní funkce. Bezpečnostní rámec ASP.NET obsahuje třídy pro autentizaci a autorizaci uživatelů a také další funkce podporující bezpečnostní opatření. S použitím těchto funkcí a také obecných technik pro zajištění bezpečnosti aplikace jsme schopni vybudovat webový informační systém, který bude dostatečně bezpečný z hlediska běžného použití. Detaily týkající se komplexního zajištění bezpečnosti aplikace v ASP.NET najdete v odpovídající literatuře (viz. [ASP.NET 3]).

Následující seznam bezpečnostních doporučení a technologií by měl být uvažován při následné implementaci webového IS. Jde především o pečlivé ověřování uživatelského vstupu, eliminaci škodlivých SQL dotazů (tzv. SQL injenction), omezení ukládání citlivých dat do skrytých polí webové stránky (hidden fields) nebo do stavu zobrazení (ViewState). Při použití základní nebo formulářové autentizace by mělo být zapnuto SSL (Secure Socket Layer). Použijeme-li formulářovou autentizaci, tak bychom měli ochránit autentizací cookies, tím, že nastavíme časové limity co nejkratší.

ASP.NET obsahuje základní infrastrukturu pro vykonávání autentizace a autorizace. Základní knihovna tříd .NET Frameworku obsahuje některé třídy jmenného prostoru Systém.Security pro šifrování a podepisování dat. Pro přístup do aplikace HaPJP bude třeba implementovat právě autentizaci i autorizaci uživatelů. Další bezpečnostní opatření se týká přímo nastavení IIS, na kterém bude aplikace nasazena (viz. [ASP.NET 1], [ASP.NET 3]).

4.2.8 Výkon

Poslední částí kapitoly Návrh implementace (viz. 4) bude zmínka o výkonu aplikace a faktorech, které samotný výkon ovlivňují. Jelikož se jedná o webovou aplikaci, tak na výkon má vliv především rychlost internetového připojení. Naše aplikace HaPJP by měla být nasazena na serveru v síti VŠB-TU Ostrava a pro uživatele z řad zaměstnanců planetária by tak měla být dostatečně rychlá. Navíc ze specifikace požadavků vychází skutečnost, že se nepředpokládá velké množství paralelně pracujících uživatelů se systémem. Nicméně nezbytný přenos dat mezi klientem a serverem pomocí protokolu HTTP můžeme optimalizovat pomocí validace na straně klienta (použitím Javascriptu) nebo použitím technologie AJAX. Z hlediska ASP.NET můžeme použít technologii ukládání do paměti cache, která je určena k dočasnému uchovávání kopií informací.

Dalšími faktory, které zde mohou ovlivnit výkon, jsou samotná struktura databáze, resp. návrh indexů, a použité SQL dotazy, resp. uložené procedury v MS SQL Serveru 2005. Pro optimalizaci výkonu aplikace bychom měli důkladně uvážit použití indexů, které sice urychlují vyhledávání a třídění dat, ale velké množství trvalých indexů, které jsou zbytečné, naopak aplikaci zpomaluje. Co se týká SQL dotazů, tak i zde platí, že špatně napsaný SQL dotaz může aplikaci zpomalovat a mělo by proto dojít k optimalizaci samotných SQL dotazů a uložených procedur.

5. Implementace

Implementace informačního systému vychází z již provedeného návrhu implementace, kde bylo rozhodnuto o použitém prostředí a kde byly také zpracovány další podklady pro samotnou implementaci, který by už měla být rutinní záležitostí. Součástí implementace je také vytvoření dokumentace a testování systému.

Nejdříve jsem musel v MS SQL Serveru definovat novou databázi a všechny její tabulky včetně relací a dalších deklarovaných integritních omezení. Potom jsem mohl přistoupit k samotné implementaci informačního systému v .NET. Při implementaci jsem oddělil databázovou logiku do samostatné assembly v C#, pomocí které se přistupuje do databáze. Stejná assembly obsahuje také objektový model, tj. třídy v C# reprezentující jednotlivé databázové tabulky. Pomocí této assembly se volají uložené procedury na straně MS SQL Serveru. Uložené procedury (stored procedures) jsou realizovány v jazyku T-SQL (Transact-SQL). Další částí implementace je potom samostatná webová aplikace využívající právě předchozí assembly. Oddělení jednotlivých částí aplikace vychází z architektury MVC (Model-View-Controller), která odděluje datovou logiku aplikace, uživatelské rozhraní a řídící logiku. Samotná webová aplikace je implementována v ASP.NET a využívá další technologie jako CSS pro definici stylů zobrazení webové stránky v (X)HTML (viz. [CSS a XHTML]) nebo AJAX (viz. [AJAX]), což je technologie vývoje interaktivních webových aplikací, které mění obsah svých stránek dynamicky bez nutnosti odesílání celého obsahu na server a zpět. Pro použití v .NET jsem použil freeware knihovnu AJAX.NET (viz. [AJAX.NET]), která integruje technologie AJAX a platformu .NET. Technologie AJAX obecně vyžaduje podporu JavaScriptu, tzn. webové prohlížeče, ve kterých aplikace poběží, musí mít JavaScript zapnutý. Následující kapitola bude obsahovat základní ukázky implementovaných funkcí a také ukázky použitého programového kódu.

Celý systém byl úspěšně implementován a jeho testovací verze je nasazena na Hvězdárně a planetáriu Johanna Palisy. Internetová adresa, na které je aplikace dostupná, je zatím nastavena na http://158.196.88.11/HaPJP/. Aplikace ještě není dostupná pro online zákazníky, zatím se k ní dá připojit pouze ze sítě VŠB-TU Ostrava, což je dostačující pro testovací provoz zaměstnanců HaPJP. Pakliže systém bude nasazen do provozu, potom se zpřístupní i uživatelům mimo síť VŠB a pro IP adresu se nastaví nějaký alias. Pro přihlášení do aplikace použijte přihlašovací jméno (*admin*) a heslo (*?administrator*).

5.1 Uživatelské funkce

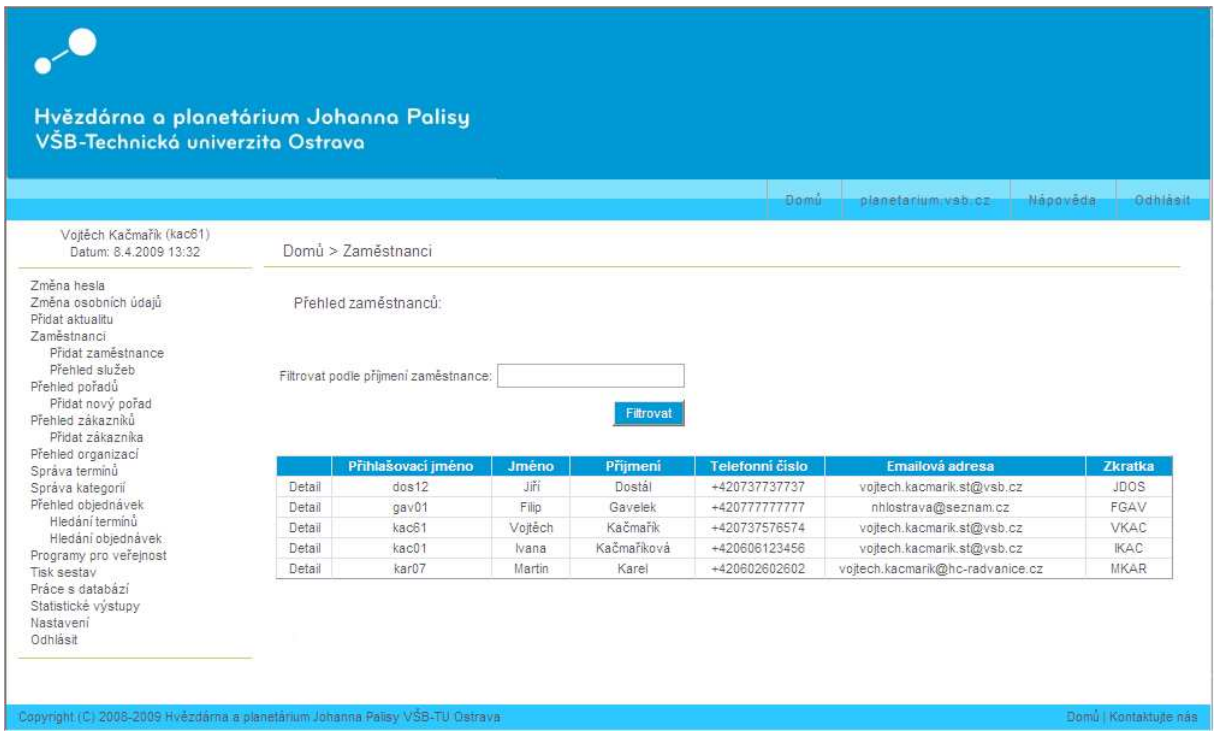
Z funkční analýzy a z návrhu implementace vycházejí všechny potřebné detaily, podle kterých jsem realizoval několik následujících uživatelských funkcí, které tvoří základ IS pro HaPJP a budou pravděpodobně nejčastěji používány.

Funkce pro přihlášení do systému, odhlášení ze systému, změnu hesla nebo pro generování nového hesla jsou realizovány s pomocí API pro členství (Membership API) ASP.NET a ovládacích prvků využívajících toto API pro členství, např. Login, LoginStatus, ChangePassword aj. Přihlašovací formulář do IS je zobrazen na následujícím obrázku (viz. Obrázek 5.1). Na následujících obrázcích je možné vidět základní obrazovku celého IS, která obsahuje záhlaví, zápatí, uživatelské menu, hlavní

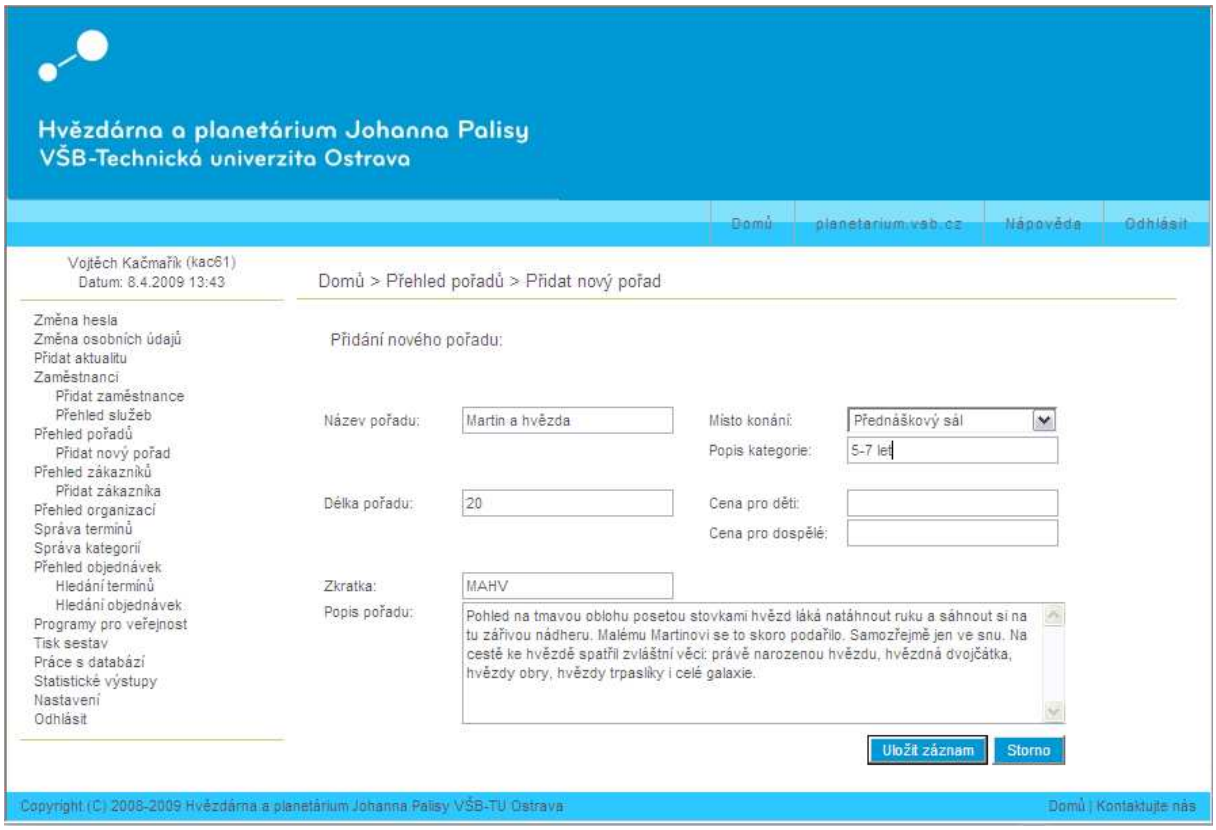
pracovní plochu, tak jak bylo navrženo v kapitole Návrh implementace (viz. 4), a několik formulářů, resp. vstupních obrazovek, které realizují základní funkce systému.

Obrázek 5.1: Přihlašovací formulář

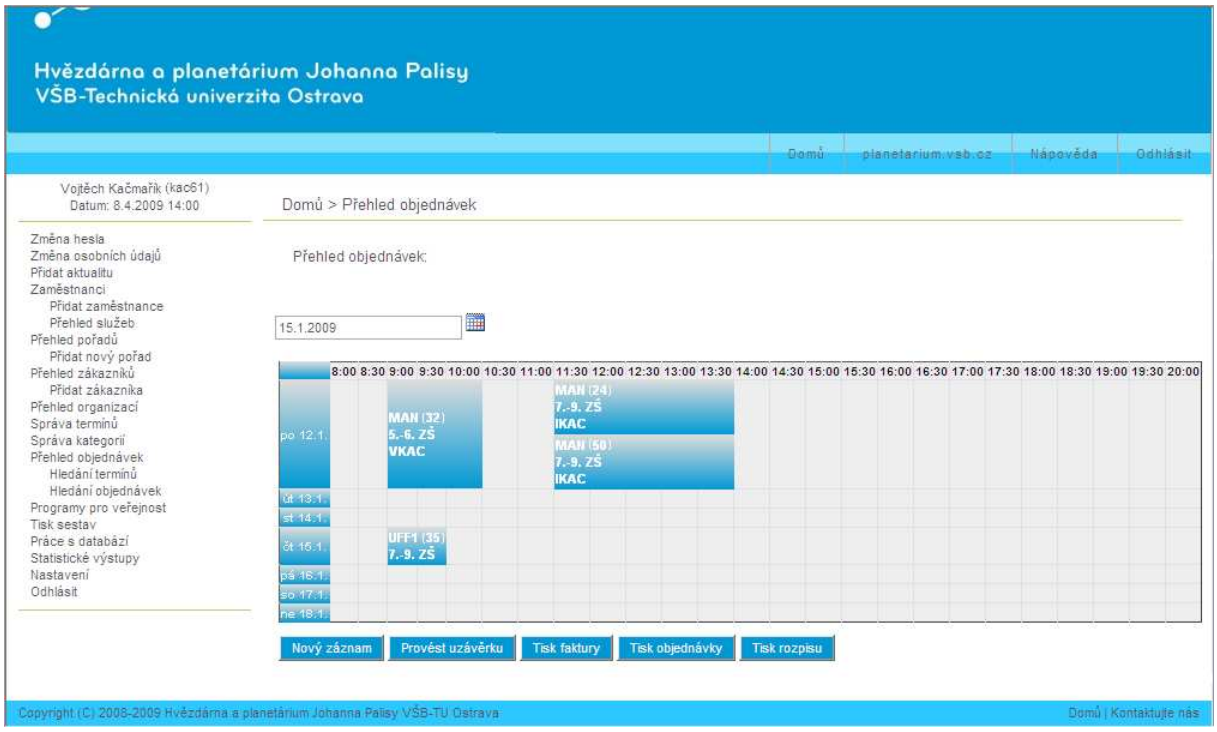
Přehled zaměstnanců (viz. Obrázek 5.2) slouží k výpisu všech záznamů o zaměstnancích HaPJP v jednoduché tabulce, která obsahuje základní evidované údaje. Je možné použít filtr a nechat si tak vypsat pouze toho zaměstnance, který nás zajímá. U každého záznamu v tabulce je navíc tlačítko pro detail zaměstnance. Při detailním výpisu můžeme záznam editovat nebo smazat. Další funkce (viz. Obrázek 5.3) slouží pro přidání nového pořadu do databáze IS. Uživatel musí vyplnit požadované vstupní hodnoty a záznam je potom uložen do databáze.



Obrázek 5.2: Přehled zaměstnanců

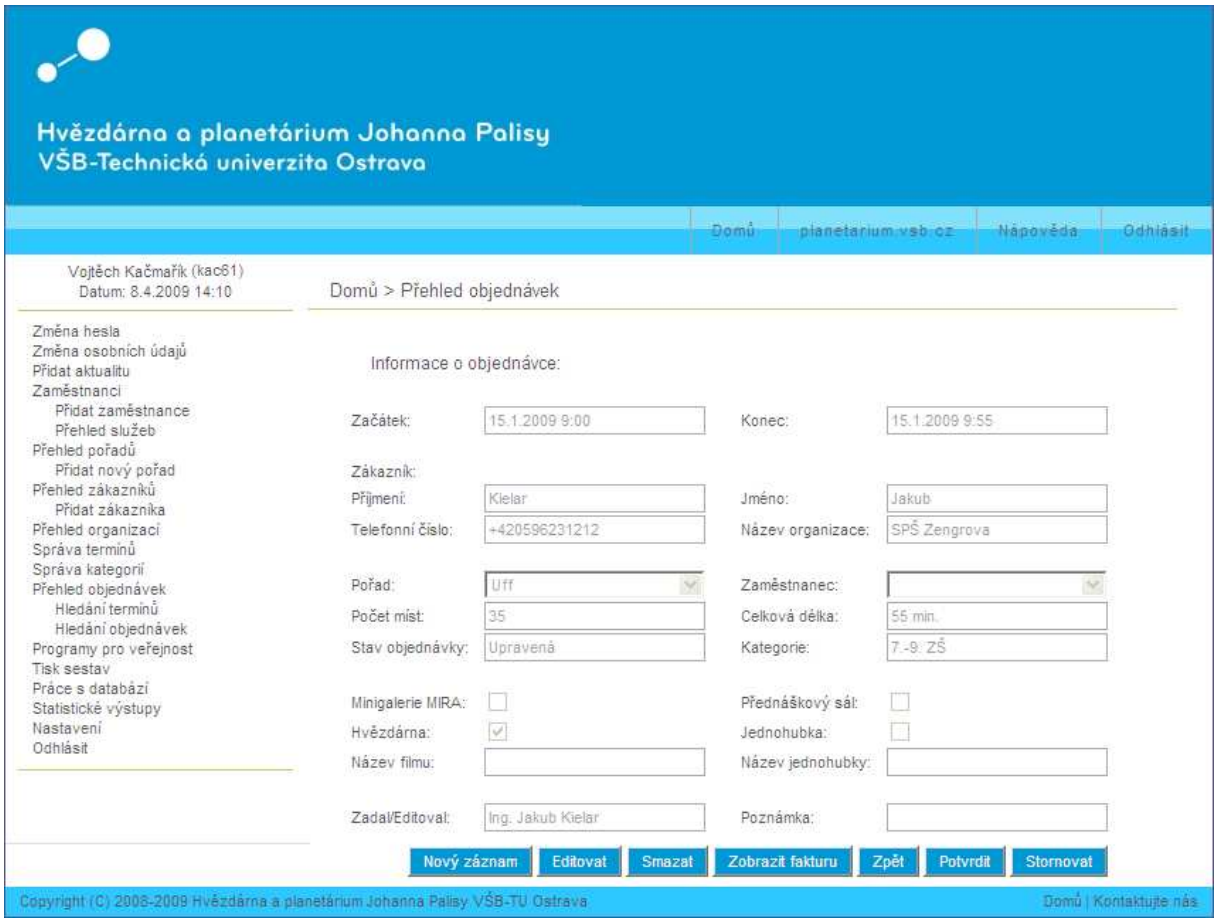


Obrázek 5.3: Přidání nového pořadu



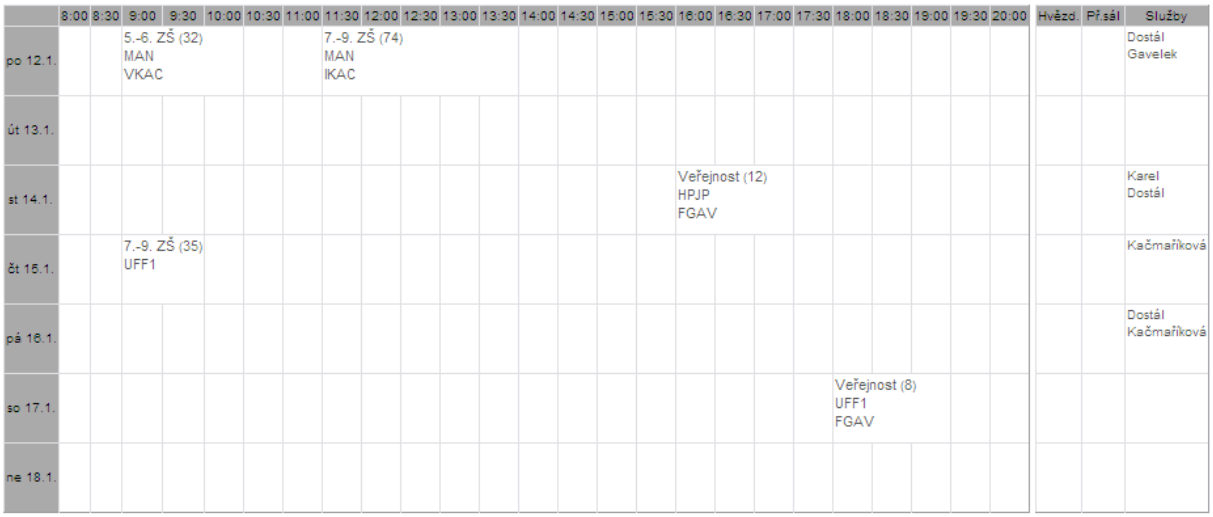
Obrázek 5.4: Přehled objednávek

Přehled objednávek (viz. Obrázek 5.4) je nejdůležitější funkce IS, která koresponduje s hlavním cílem aplikace, a časový harmonogram na předcházejícím obrázku vychází z tištěného formuláře, který se používal na HaPJP před tímto systémem. Zobrazuje objednávky ve vybraném týdnu, přičemž u každé objednávky jsou vypsány nejdůležitější informace. Pokud uživatel klikne na vybranou objednávku, tak je zobrazen detail objednávky (viz. Obrázek 5.5). Z těchto dat je potom vytvořena tisková sestava (viz. Obrázek 5.6), která je formátována pro finální tisk na tiskárně.

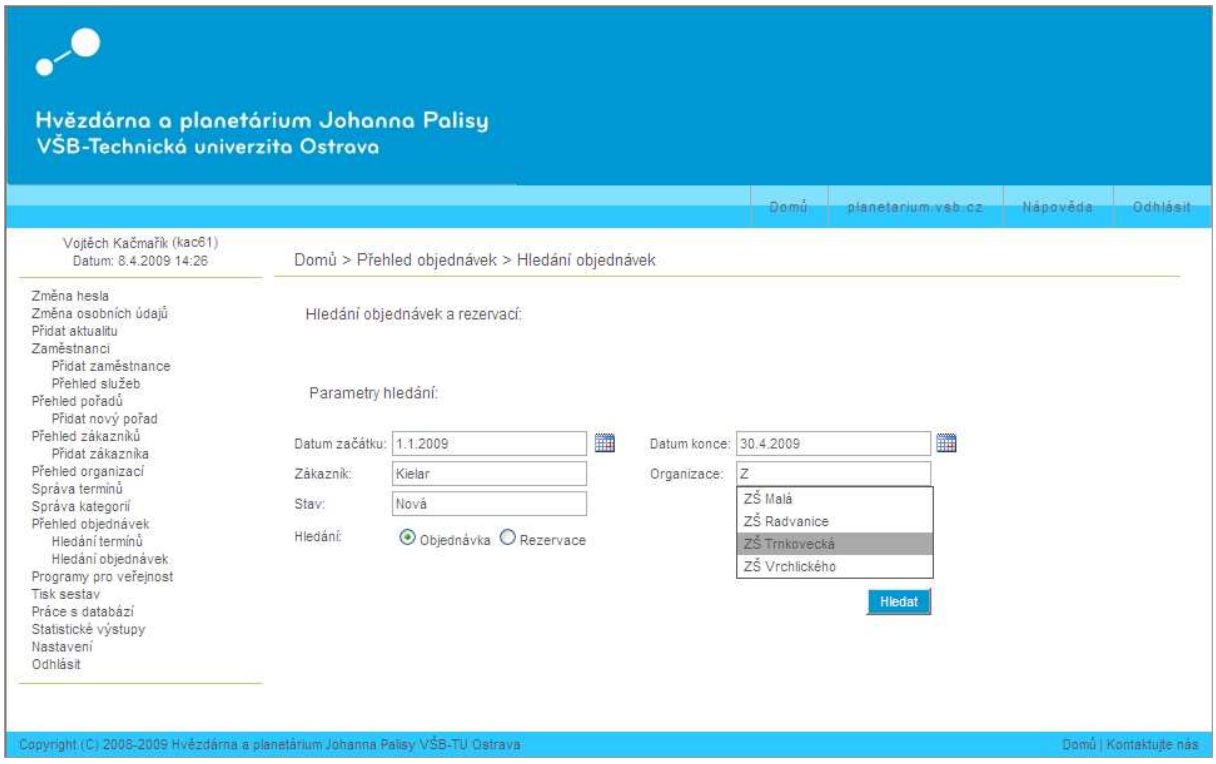


Obrázek 5.5: Objednávka

V objednávkách také samozřejmě existuje možnost vyhledávání podle určitých kritérií. Uživatel musí vybrat v menu funkci Hledání objednávek a je mu zobrazen následující formulář (viz. Obrázek 5.7). Záhlaví obsahuje kritéria hledání jako datum, jméno zákazníka, stav objednávky apod. Je zde právě využita technologie AJAX.NET, konkrétně prvky AutoComplete a CalendarExtender. Po zadání všech kritérií jsou zobrazeny určité informace o objednávce a odkaz na detail objednávky. Uživatel se tak snadno dostane na přehled objednávek, resp. na vybranou objednávku.



Obrázek 5.6: Tisk rozpisu



Obrázek 5.7: Hledání objednávek

5.2 Ukázky vlastní implementace

Pro názornou ukázku vlastní implementace jsem se rozhodl, že do textu mé diplomové práce zařadím krátké příklady demonstrující určité části nebo základní principy implementace výsledné webové aplikace v ASP.NET. Základním jazykem, který jsem použil při implementaci v .NET, je jazyk C# ve verzi 3.0.

První příklad implementace ukazuje základní použití C# pro definici nové třídy v rámci objektového návrhu, např. třída reprezentující pořad na HaPJP. Objektový návrh je umístěn v samostatné assembly zároveň s třídami pro práci s databází. Ostatní třídy v rámci objektového návrhu mají stejnou strukturu a syntaxi. Příklad pro účel demonstrace použití neukazuje kompletní definici této třídy.

```
namespace HaPJP.DatabaseModel.ObjectModel
{

    public class Lecture
    {
        #region Fields

        private string _name;
        private string _description;

        #endregion

        #region Constructors

        public Lecture()
        {
        }

        public Lecture(string name, string description)
        {
            this._name = name;
            this._description = description;
        }

        #endregion

        #region Properties

        public string Name
        {
            get { return this._name; }
            set { this._name = value; }
        }

        public string Description
        {
            get { return this._description; }
            set { this._description = value; }
        }

        #endregion
    }
}
```

Další krátký příklad ukazuje třídu pro práci s databází MS SQL Server. Jedná se o pomocnou třídu, která spravuje dotazy do databáze týkající se konkrétního objektu, např. objednávky apod. Uvedený příklad opět není úplný, i tak demonstruje použití tříd SqlConnection, SqlCommand a hlavně SqlTransaction pro práci s určitým objektem.

```
namespace HaPJP.DatabaseModel
{
    using System;
    using System.Data;
    using System.Data.SqlClient;
    using System.Web.Configuration;
```

```
    using HaPJP.DatabaseModel.ObjectModel;

    public class OrderDB
    {
        #region Fields

        private string _connectionString;

        #endregion

        #region Constructors

        public OrderDB()
        {
            _connectionString =
WebConfigurationManager.ConnectionStrings["planetariumConnectionString"].Connection
String;
        }

        #endregion


        #region Public Methods

        public int InsertOrder(Order order)
        {
            SqlConnection connection = new SqlConnection(_connectionString);
            SqlCommand command = new SqlCommand("InsertOrder", connection);
            command.CommandType = CommandType.StoredProcedure;

            SqlTransaction sqlTransaction = null;

            command.Parameters.Add(new SqlParameter("@count", SqlDbType.Int, 4));
            command.Parameters["@count"].Value = order.Count;

            command.Parameters.Add(new SqlParameter("@id", SqlDbType.Int, 4));
            command.Parameters["@id"].Direction = ParameterDirection.Output;

            try
            {
                connection.Open();
                sqlTransaction = connection.BeginTransaction();

                command.Transaction = sqlTransaction;
                command.ExecuteNonQuery();

                order.ID = (int)command.Parameters["@id"].Value;

                command = new SqlCommand("InsertInvoice", _connection);
                command.CommandType = CommandType.StoredProcedure;

                command.Transaction = sqlTransaction;
                command.ExecuteNonQuery();

                sqlTransaction.Commit();

                return order.ID;
            }
            catch
            {
                sqlTransaction.Rollback();
                return -1;
            }
            finally
```

```
            {
                connection.Close();
            }
        }

        #endregion
    }
}
```

Dalším příkladem je ukázka implementace třídy Logger, která slouží pro zápis do logu a její implementace byla řešena s použitím návrhového vzoru Singleton, což znamená, že bude existovat pouze jediná instance této třídy v celé aplikaci.

```
namespace HaPJP.Common
{
    using System;
    using System.Configuration;
    using System.Diagnostics;
    using System.Globalization;
    using System.IO;
    using System.Threading;

    /// <summary>
    /// Poskytuje funkcionalitu pro zápis zpráv do logovacího souboru
    /// </summary>
    public class Logger
    {
        #region Members

        private static Logger _instance;

        private string _logPath;
        private string _logName;

        #endregion

        #region Constructors

        private Logger()
        {
            this.InitializeSettings();
        }

        #endregion

        #region Static Property

        public static Logger Instance
        {
            get
            {
                if (_instance == null)
                    _instance = new Logger();

                return _instance;
            }
        }

        #endregion

        /// <summary>
        /// Metoda pro zápis zprávy do logovacího souboru
        /// </summary>
```

```
        /// <param name="message">Zpráva k zápisu</param>
        /// <param name="type">Typ události</param>
        public void WriteToLog(string message, EventLogEntryType type)
        {
            // implementace ve zdrojovém textu
        }

        /// <summary>
        /// Nastaví proměnné podle nastavení z konfiguračního souboru
        /// </summary>
        private void InitializeSettings()
        {
            this._logPath =
ConfigurationManager.AppSettings["HaPJP.Logging.LogPath"].ToString();
            this._logName =
ConfigurationManager.AppSettings["HaPJP.Logging.LogName"].ToString();
        }
    }
}
```

Na dalším příkladu bych chtěl ukázat použití T-SQL při implementaci uložených procedur na straně MS SQL Serveru. Jednoznačná výhoda použití uložených procedur je oddělení databázové logiky od vlastní implementace webové aplikace v ASP.NET. Nevýhodou by pak mohla být horší údržba a ladění uložených procedur.

```
CREATE PROCEDURE DeleteOrder(@id int)
AS
BEGIN
BEGIN TRY
     BEGIN TRANSACTION    -- Začátek transakce, tj. BEGIN

     DECLARE @dateId int;
     DECLARE @count int;

     SET @dateId = (SELECT [id_termin]
                           FROM dbo.[Objednavka]
                           WHERE [id_objednavka] = @id)

     -- Zjištění, jestli na tento termín existují také jiné objednávky
     SET @count = (SELECT COUNT(*)
                           FROM dbo.[Objednavka]
                           WHERE [id_termin] = @dateId
                                AND [id_objednavka] != @id)

     -- Odstranění záznamu o objednávce z databáze
     DELETE
          FROM dbo.[Objednavka]
          WHERE [id_objednavka] = @id

     IF @count = 0
          BEGIN
               -- Odstranění záznamu o termínu z databáze
               DELETE
                    FROM dbo.[Termin]
                    WHERE [id_termin] = @dateId
          END

     -- Jestliže se vyskytla nějaká chyba, potom ROLLBACK transakce
     IF @@ERROR <> 0
          ROLLBACK TRANSACTION

     -- Jestliže jsme se dostali až zde, potom COMMIT transakce
     COMMIT TRANSACTION
```

```
END TRY
BEGIN CATCH
     -- Jestliže se vyskytla neočekávaná chyba, potom ROLLBACK za předpokladu, že
existuje nějaká transakce
     IF @@TRANCOUNT > 0
          ROLLBACK

     -- Vyvolá error s patřičnou zprávou atd.
     DECLARE @ErrMsg nvarchar(4000), @ErrSeverity int
     SELECT @ErrMsg = ERROR_MESSAGE(), @ErrSeverity = ERROR_SEVERITY()

     RAISERROR(@ErrMsg, @ErrSeverity, 1)
END CATCH
END
```

Další části webové aplikace, jako samotné webové stránky v ASP.NET a tzv. code behind, neboli kód na pozadí, jsou natolik specifické, že konkrétní příklady z této části implementace tady nebudu pro přehlednost uvádět. Samotné zdrojové texty jsou k dispozici na přiloženém CD k této práci v adresáři *Source_code\WebSites\HaPJP* (viz. Obsah přiloženého CD) a jsou také nasazeny na testovacím prostředí na Hvězdárně a planetáriu Johanna Palisy.

5.3 Testování systému

Důležitá část vlastní implementace je testování aplikace, které slouží k nalezení chyb a jejich následnému odstranění. Pro testování webové aplikace jsem využil nástroje Manual Tester, Functional Tester a Performance Tester od firmy IBM Rational (viz. [IBM]). Jak je z názvů nástrojů patrné, tak provedené testování bylo zaměřeno na manuální testy, funkční testy a výkonnostní testy. Manuální testy sloužily k testování vzhledu aplikace, zobrazení menu a dalších tlačítek, zarovnání tabulek, formulářů apod. Funkční testy byly potom zaměřeny na testování vlastní funkcionality aplikace, tj. byly provedeny všechny funkce systému se správnými i chybnými vstupními daty a testování bylo zaměřeno na chování aplikace. Po provedení manuálních a funkční testů jsem všechny nalezené chyby odstranil a implementovaná aplikace by tak měla být připravena pro testovací provoz, který pravděpodobně odhalí další chyby nebo připomínky uživatelů.

Poslední testy, které jsem na aplikaci prováděl, byly testy výkonnostní. Tyto testy jsou zaměřeny na výkon webové aplikace, tj. na uživatelskou odezvu v závislosti na počtu paralelních uživatelů webové aplikace, rychlost databázových dotazů a propustnosti (rychlosti) připojení k Internetu. Testování se provádí simulováním provádění běžných uživatelských funkcí paralelně několika uživateli a výsledky zobrazují výkon aplikace. Po provedení testů bylo z dosažených výsledků patrné, že aplikace je pro běžný provoz (předpokládané zatížení) dostatečně výkonná a mohla by být nasazena do provozu.

5.4 Konfigurace a nasazení

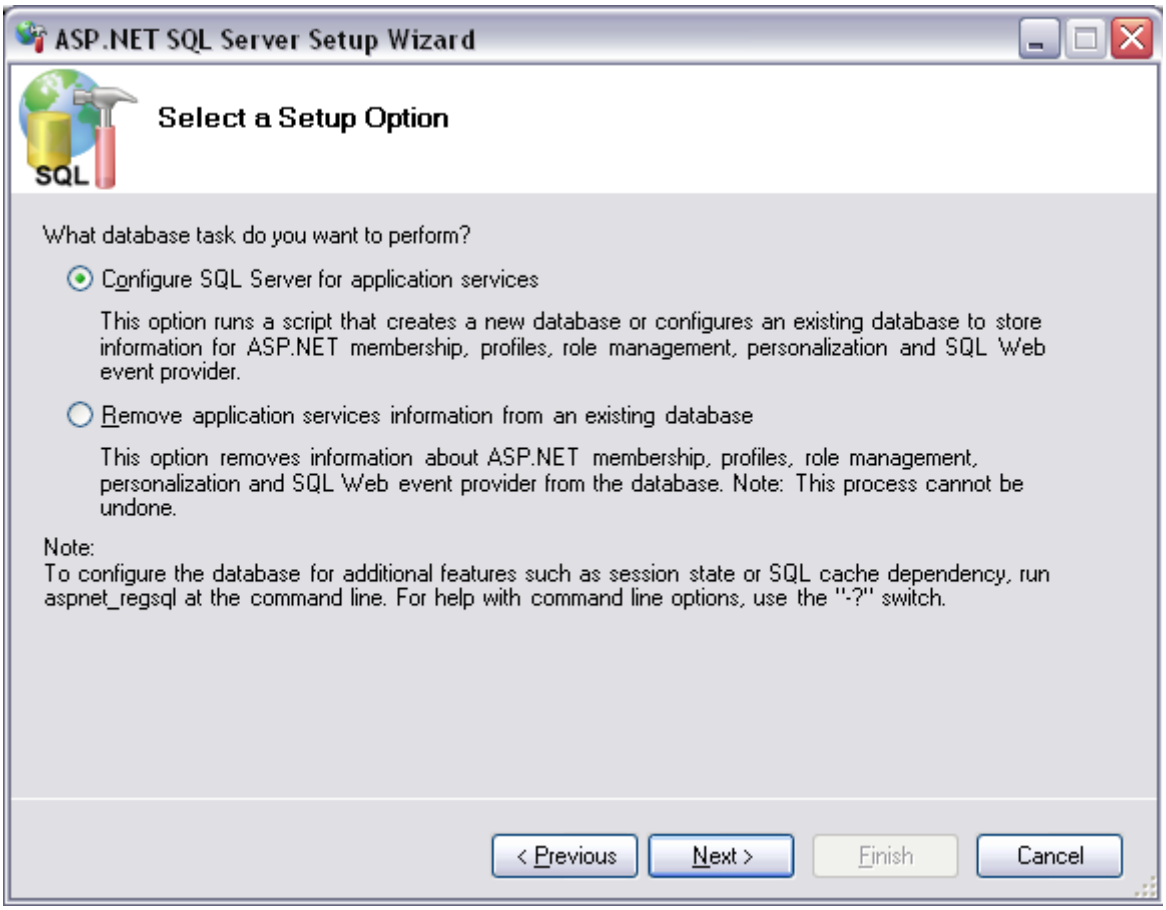
Konfigurace a nasazení je nedílnou součástí vývoje libovolné aplikace, neboť právě nasazení na testovací nebo provozní prostředí a správná konfigurace tohoto prostředí je nezbytně nutné pro běh samotné aplikace u zákazníka.

Jelikož naše aplikace je webový informační systém v ASP.NET, budu se zabývat konfigurací a nasazením pro takový IS. Jako první je nutné nainstalovat na provozní počítač webový server IIS

(dodáván spolu s Windows) a také .NET Framework. Pokud instalujeme IIS 6.0 je třeba při instalaci zapnout podporu ASP.NET. Po instalaci IIS se na systémovém disku počítače vytvoří adresář s názvem *InetPub\wwwroot*, do kterého se ukládají všechny stránky, které mají být umístěny na serveru. Je nutné, abychom prostřednictvím IIS vytvořili nový virtuální adresář *HaPJP*, do kterého musíme umístit zdrojové texty naší aplikace z adresáře *Source_code\WebSites\HaPJP* (viz. Obsah přiloženého CD). Tento virtuální adresář bude odpovídat fyzickému adresáři *InetPub\wwwroot\HaPJP* přímo na systémovém disku. Při vytváření virtuálního adresáře je nutné nastavit přístupová práva Read a Run scripts (such as ASP). Pokud na cílovém stroji nejsou nainstalovány použité globální assembly, musíme je nainstalovat pomocí utility příkazového řádku .NET Frameworku, tj. gacutil.exe. Navíc je třeba nakonfigurovat IIS, tak jak to aplikace ASP.NET vyžaduje (viz. [ASP.NET 1], [ASP.NET 3]). Pro provoz webové aplikace v Internetu bude třeba zřejmě povolit ve firewallu TCP/IP komunikaci na portech 80 (HTTP) a 443 (HTTPS).

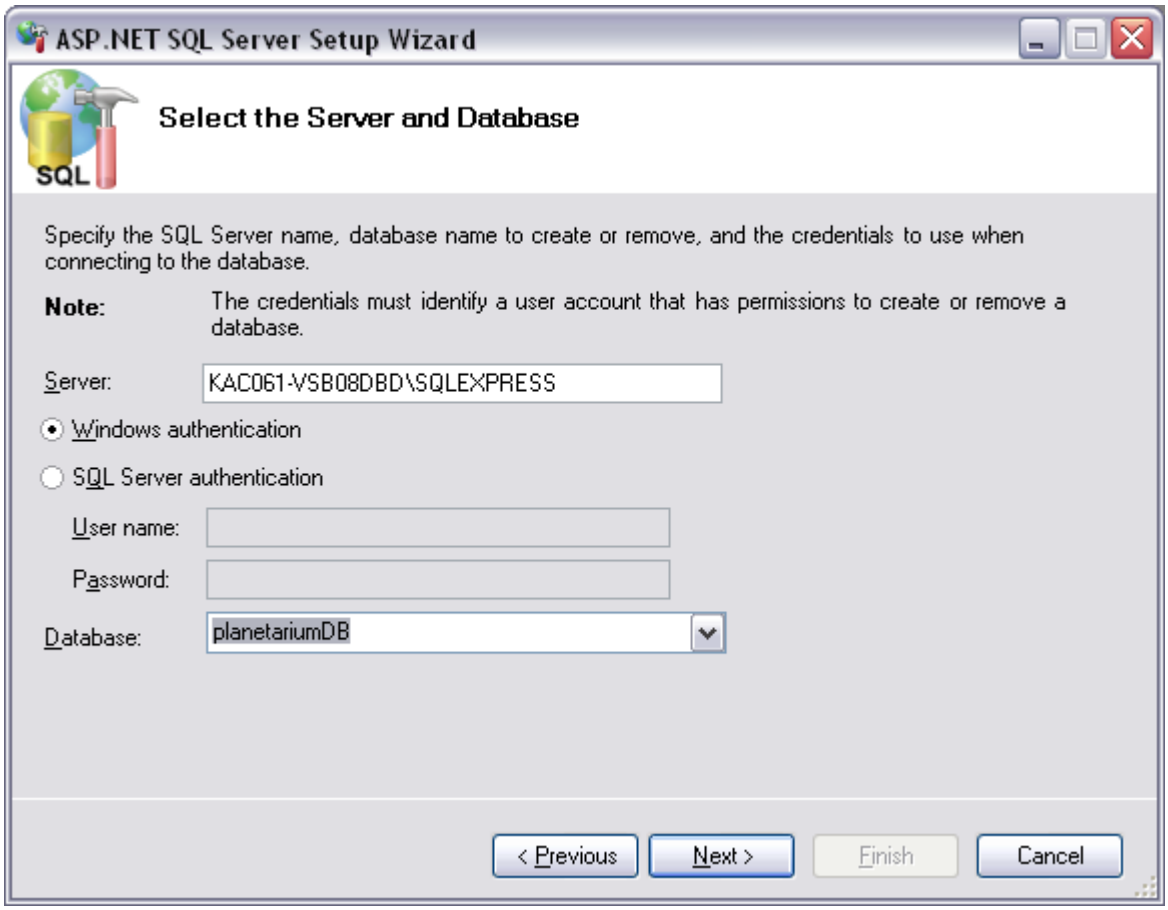
Dalším krokem je vytvoření domovského adresáře *HaPJP* přímo na systémovém disku. Obsah tohoto adresáře je kopií adresáře *Home_directory\HaPJP* na přiloženém CD. Důležité adresáře pro nasazení aplikace jsou adresáře *Install* a *Backup*. Obsah adresáře *Install* je důležitý pro vytvoření struktury databáze, vytvoření základních dat pro provoz a také vytvoření uložených procedur na MS SQL Serveru. V adresáři *Backup* je pouze jeden spustitelný soubor *backup.bat*, který provádí zálohu databáze, a jeho spouštění by mělo být přidáno do naplánovaných úloh OS Windows. Tento skript je třeba modifikovat a nastavit zde proměnnou *ServerName*.

Když máme vytvořen domovský adresář, můžeme přistoupit k dalšímu velmi důležitému kroku, tj. vytvoření a nakonfigurování databáze aplikace. Je důležité vytvořit nejenom databázi a její tabulky, ale také nakonfigurovat přihlašování k databázovému serveru a uživatele databáze. Adresář *Install* obsahuje SQL skripty a instalační soubor *install.bat*, který slouží k vytvoření databázové struktury atd. Také tento skript je třeba modifikovat a nastavit zde proměnnou *ServerName*. Potom můžeme provést instalaci z příkazové řádky. Potom, co nainstalujeme databázi, musíme tuto databázi rozšířit o tabulky potřebné pro Membership a Roles API ASP.NET. Slouží k tomu utilita *aspnet_regsql.exe*, která se nachází v adresáři *C:\WINDOWS\Microsoft.NET\Framework\v2.0.50727*, a můžeme ji spustit z příkazové řádky. Po spuštění provedeme nastavení podle následujících obrázků (viz. Obrázek 5.8, Obrázek 5.9). První okno nastavuje typ operace, jakou chceme provádět. Zvolíme zde možnost *Configure SQL Server for application services*.



Obrázek 5.8: ASP.NET SQL Server Setup Wizard part 1

Další okno nastavuje jméno databázového serveru, který bude použit pro vytvoření tabulek a uložených procedur pro Membership API, typ autentizace a jméno konkrétní databáze. Tady musíme nastavit jméno serveru podle skutečné hodnoty na provozním prostředí. Potom můžeme zvolit defaultní databázi *planetariumDB*.



Obrázek 5.9: ASP.NET SQL Server Setup Wizard part 2

Zbývá nám nastavit přihlašování k databázovému serveru atd. Pro toto nastavení můžeme využít nástroj SQL Management Studio Express, který se instaluje zároveň s MS SQL Serverem. Je třeba nastavit *Databases → Security → Logins → New Login* a zde zadat *Login Name*, *SQL Server authentization*, *Password*, zaškrtnout *Enforce password policy* a dále nastavit *Server Roles* pro tohoto uživatele a *User Mapping* na naši databázi *planetariumDB*. Potom je třeba přidat přístup pro tohoto uživatele na naši databázi. Proveďte *Databases → planetariumDB → Security → Users → New User → Login Name → Browse* a vyhledejte přidaného uživatele podle přihlašovacího jména. Zde jde nastavit *Default Schema*, *Owned schema* a *Database role membership*. To je třeba nastavit podle potřeby našeho uživatele, který bude k webové aplikaci přistupovat (tj. práva *datareader*, *datawriter*, apod.). Uživateli bychom neměli v žádném případě nastavovat práva jako *db_owner* nebo *db_securityadmin*. Posledním krokem je přiřadit takto vytvořeného uživatele přímo k naší databázi, tj. následující postup *Databases → planetariumDB → Properties → Permissions → User or roles → Add → Browse* a vyhledat našeho uživatele. Přihlašovací jméno a heslo tohoto databázového uživatele musíme potom nastavit do tagu *connectionStrings* v souboru web.config.

Pro nastavení aplikace v ASP.NET se používá právě konfigurační soubor web.config, který se nachází v hlavním adresáři celé aplikace. Je třeba nastavit připojovací řetězec v tagu *connectionStrings*, dále potom *mailSettings* pro emailovou komunikaci a některé aplikační proměnné v *appSettings* (např. jméno serveru apod.). Další nastavení, které se může v souboru web.config

změnit, je například *clientTarget* pro definici prohlížečů nebo *globalization*, kde se nastavuje informace o kultuře běhového prostředí (např. culture="cs-CZ"). Nesmíme zapomenout nastavit v sekci *compilation* atribut debug="false".

Pokud chceme aplikaci spustit, můžeme použít adresu http://localhost/HaPJP/ z místního počítače. Z jiných počítačů v síti musíme použít odpovídající IP adresu našeho provozního serveru (např. http://158.196.88.11/HaPJP/). Administrátor sítě VŠB-TU Ostrava by mohl také vytvořit nějaký alias pro naši IP adresu, aby výsledná webová adresa byla uživatelsky přívětivější.

5.5 Dokumentace

Součástí samotné implementace IS je tvorba potřebné dokumentace. Pro potřeby uživatelů byla vytvořena uživatelská příručka, která obsahuje popis všech funkcí systému včetně náhledů obrazovek a popisu jednotlivých vstupů každé funkce. Uživatelská dokumentace je přiložena v elektronické podobě na CD k této diplomové práci (viz. Obsah přiloženého CD). Zároveň byla vytvořena programátorská dokumentace, která je důležitá z technického pohledu pro programátory. Její další využití může být důležité při potřebě zavést do systému nějaké změny nebo přidání zcela nové funkcionality. Programátorská dokumentace obsahuje popis jednotlivých částí webové aplikace, krátký výčet použitých technologií a popis jednotlivých funkcí a struktury aplikace. Je také zpracována v elektronické podobě a uložena na přiloženém CD (viz. Obsah přiloženého CD).

6. Závěr

Cílem této diplomové práce bylo podrobně analyzovat a implementovat webový informační systém, který by měl sloužit Hvězdárně a planetáriu Johanna Palisy VŠB-TU Ostrava. Ze specifikace požadavků se provedla detailní analýza (jak datová, tak funkční) a následně návrh implementace, ve kterém se mohly projevit moderní technologie dnešní doby pro tvorbu webových informačních systémů. Nakonec byl podle předcházejících kapitol systém implementován a také důsledně testován. Na testování systému se podíleli i samotní zaměstnanci HaPJP, kde je nyní systém nasazen v testovacím režimu, který je dostupný na adrese http://158.196.88.11/HaPJP/. Celý systém by měl být v budoucnu používán v praxi. Pro potřeby uživatelů, případně programátorů, byla navíc vytvořena podrobná uživatelská a programátorská dokumentace.

Při realizaci této diplomové práce jsem musel využít všech svých znalostí a také dalších informačních zdrojů, týkajících se dané problematiky a zvolených technologií. Informační systém řeší celkovou agendu na Hvězdárně a planetáriu Johanna Palisy a je tak poměrně hodně specifický. Některé funkce byly hodně detailně specifikovány a obsahují různé detaily. Na první pohled by se mohlo zdát, že se jedná o objednávkový systém. Ale i při realizaci objednávek se v systému vyskytují specifické detaily, které bylo třeba vyřešit. Další funkce poskytují možnost spravovat i ostatní údaje, které bezprostředně souvisejí s provozem HaPJP. Systém byl dělán na zakázku a nedá se s jistotou říci, zda by jej mohly využívat i další hvězdárny. Jak už bylo řečeno, mnohá funkcionalita je velmi specifická a jestliže by měl být IS použit i pro další hvězdárnu, tak by musely být realizovány zřejmě další úpravy. Princip objednávek, které jsou v systému nejdůležitější, by možná mohl zůstat zachován.

Všechny specifikované funkce a detaily se mi podařilo v navrženém prostředí vyřešit. Použitím technologií ASP.NET a CSS jsem vytvořil webový informační systém, který odpovídá dnešním standardům pro webové aplikace. Z hlediska dalšího vývoje by do informačního systému mohla být přidána další funkcionalita, např. větší počet on-line funkcí pro zákazníky, lepší evidence faktur a plateb, nebo rozšíření tiskových sestav apod. Celý projekt webové aplikace je podrobně dokumentován a díky použitým technologiím i vývojovému prostředí je celkem přehledný. Na základě požadavků zadavatele by mohl v budoucnu obsahovat navíc i další funkce.

Literatura

[Šarm 1] ŠARMANOVÁ, Jana. *Databázové a informační systémy : Sylaby přednášek FEI VŠB-TU Ostrava*. Ostrava : VŠB-Technická univerzita Ostrava, 2005.

[Šarm 2] ŠARMANOVÁ, Jana. *Informační systémy a datové sklady : Sylaby přednášek FEI VŠB-TU Ostrava*. Ostrava : VŠB-Technická univerzita Ostrava, 2006.

[Šarm 3] ŠARMANOVÁ, Jana. *Teorie zpracování dat : Skriptum FEI VŠB-TU Ostrava*. Ostrava : VŠB-Technická univerzita Ostrava, 1997.

[Šarm 4] ŠARMANOVÁ, Jana. *Databázové a informační systémy :Učební text FEI VŠB-TU Ostrava*. Ostrava : VŠB-Technická univerzita Ostrava, 2007. ISBN 978-80-248-1499-5.

[MS SQL 2005] Solid Quality Learning. *Microsoft SQL Server 2005 : Základy databází - Krok za krokem*. Brno : Computer Press, a.s., 2007. ISBN 978-80-251-1524-4.

[ASP.NET 1] AVERY, James. *Microsoft ASP.NET Konfigurace a nastavení : Kapesní rádce programátora a administrátora*. Brno : Computer Press, a.s., 2004. ISBN 80-251-0121-5.

[ASP.NET 2] PÍSEK, Slavoj. *ASP.NET Začínáme programovat : Podrobný průvodce začínajícího uživatele*. Praha : Grada Publishing a.s., 2003. ISBN 80-247-0526-5.

[ASP.NET 3] MACDONALD, Matthew; SZPUSZTA, Mario. *ASP.NET 2.0 a C# : tvorba dynamických stránek PROFESIONÁLNĚ*. Brno : Zoner Press, 2006. ISBN 80-86815-38-2.

[C# 1] GUNNERSON, Eric. *Začínáme programovat v C#*. Praha : Computer Press, 2001. ISBN 80-7226-525-3.

[C# 2] TROELSEN, Andrew. *C# a .NET 2.0 PROFESIONÁLNĚ*. Brno : Zoner Press, 2006. ISBN 80-86815-42-0.

[CSS a XHTML] DRUSKA, Peter. *CSS a XHTML : Tvorba dokonalých webových stránek krok za krokem*. Praha : Grada Publishing a.s., 2006. ISBN 80-247-1382-9.

[AJAX] ASLESON, Ryan; SCHUTTA, Nathaniel T. *AJAX : Vytváříme vysoce interaktivní webové aplikace*. Brno : Computer Press, a.s., 2006. ISBN 80-251-1285-3.

[GoF] GAMMA, Erich; HELM, Richard; JOHNSON, Ralph; VLISSIDES, John. *Návrh programů pomocí vzorů*. Praha : Grada Publishing a.s., 2003. ISBN 80-247-0302-5.

[Singleton] Microsoft Corporation. *Implementing Singleton in C#* [online]. 2009. Dostupné z: <http://msdn.microsoft.com/en-us/library/ms998558.aspx>

[MSDN] Microsoft Corporation. *MSDN Library* [online]. 2009. Dostupné z: http://msdn.microsoft.com/en-us/library/default.aspx

[AJAX.NET] Microsoft Corporation. *AJAX : The Official Microsoft ASP.NET Site* [online]. 2009. Dostupné z: http://www.asp.net/ajax/

[IBM] International Business Machines Corp. *IBM Rational Software* [online]. 2009. Dostupné z: http://www-01.ibm.com/software/rational/

Přílohy

A Vzorové formuláře pro tisk

	8^{30}	10^{00}	11^{30}	13^{00}	14^{00}	16^{00}	18^{00}	hvězd.	př.sál	služby
Po 26/L	1.-5. (50) PZH JH		3 ŠTO (40) VŽ JH							MV
Út 27/L	9.r. (40) ZŠKV MV	9.r. (40) ZŠKV MV								~~MV~~ PK
St 28/L	MŠ (60) M*, MVK MO	4.r. (35) PZH MO				VEL. VŽ TG	VEL. VERNIŠÁŽ TG	VEL. TG		JH KVĚTÁKOVÁ DLUHOŠ
Čt 1/3	5.r. (33) VT PK	1.-2. (40) PZH PK								IM
Pá 2/3										MO
So 3/3					VEL. PZH IM	VEL. SKV IM	VEL. HUDBA IM	VEL. ZATMĚNÍ (? MV		HLAVÁČ, KONEČNÝ LEXA HJNEK
Ne 4/3										

PLANETÁRIUM						PŘEDNÁŠKOVÝ SÁL			HVĚZDÁRNA
hod.	škola	jméno, telefon	třída	počet	pořad	hod.	škola	film	

Datum	Hod.	Škola	PSČ	Třída	Počet	Planetárium pořad		Planetárium jednohubky		Přednáškový sál filmy		MIRA		Hvězdárna pro školy	
						děti	dospělí	děti	dospělí	děti	dospělí	děti	dospělí	děti	dospělí

B Datový slovník

Tabulka	Atribut	Datový typ	Délka	Klíč	Null	Index	IO	Význam, poznámka
Aktualita	id_aktualita	int		A	N			identity(1,1)
	id_zamestnanec	varchar	5	N	N			cizí klíč Zamestnanec
	datum	datetime		N	N	A	*1	datum vložení
	text	varchar	500	N	N			obsah aktuality

*1 … datum vložení musí být následujícího formátu: dd.MM.yyyy hh:mm:ss

Tabulka	Atribut	Datový typ	Délka	Klíč	Null	Index	IO	Význam, poznámka
Faktura	id_faktura	int		A	N			identity(1,1)
	potvrzeno	bit	1	N	N		*1	je potvrzená?
	cena_celkem	int		N	A			celková cena
	celkovy_pocet	int		N	N			celkový počet osob
	celkem_deti	int		N	N			počet dětí
	celkem_dospeli	int		N	N			počet dospělých osob
	porad_deti	int		N	A			
	porad_dospeli	int		N	A			
	jednohubka_deti	int		N	A			
	jednohubka_dospeli	int		N	A			
	film_deti	int		N	A			
	film_dospeli	int		N	A			
	mira_deti	int		N	A			
	mira_dospeli	int		N	A			
	hvezdarna_deti	int		N	A			
	hvezdarna_dospeli	int		N	A			

*1 … hodnota True/False … DEFAULT 0

Tabulka	Atribut	Datový typ	Délka	Klíč	Null	Index	IO	Význam, poznámka
Kategorie	id_kategorie	int		A	N			identity(1,1)
	nazev	varchar	30	N	N	A		název kategorie
	popis	varchar	150	N	N			popis kategorie

Tabulka	Atribut	Datový typ	Délka	Klíč	Null	Index	IO	Význam, poznámka
Komentar	id_komentar	int		A	N			identity(1,1)
	id_osoba	varchar	7	N	N			cizí klíč Osoba
	id_porad	int		N	N	A		cizí klíč Porad
	datum_komentare	datetime		N	N	A	*1	datum vložení komentáře
	text	varchar	250	N	N			obsah komentáře

*1 … datum vložení komentáře musí být následujícího formátu: dd.MM.yyyy hh:mm:ss

Tabulka	Atribut	Datový typ	Délka	Klíč	Null	Index	IO	Význam, poznámka
Misto	id_misto	tinyint		A	N			identity(1,1)
	nazev	varchar	30	N	N			název místa na HaPJP

Tabulka	Atribut	Datový typ	Délka	Klíč	Null	Index	IO	Význam, poznámka
Objednavka	id_objednavka	int		A	N			identity(1,1)
	id_osoba	varchar	7	N	N	A		cizí klíč Osoba
	id_organizace	int		N	A	A		cizí klíč Organizace
	id_faktura	int		N	A			cizí klíč Faktura
	id_termin	int		N	N	A		cizí klíč Termin
	id_porad	int		N	A	A		cizí klíč Porad
	id_kategorie	int		N	A	A		cizí klíč Kategorie
	id_zamestnanec	varchar	5	N	A	A		cizí klíč Zamestnanec
	id_stav	tinyint		N	N	A		cizí klíč Stav
	pocet	int		N	N	A	*3	počet osob
	celkova_delka	int		N	A			celková délka obj.
	mira	bit	1	N	N		*1	minigalerie MIRA?
	prednaskovy_sal	bit	1	N	N		*1	přednáškový sál?
	hvezdarna	bit	1	N	N		*1	hvězdárna?
	jednohubka	bit	1	N	N		*1	jednohubka?
	nazev_filmu	varchar	50	N	A			název filmu
	nazev_jednohubky	varchar	50	N	A			název jednohubky
	poznamka	varchar	500	N	A			poznámka k obj.
	autor	varchar	10	N	N		*2	login autora obj.

*1 … hodnota True/False … DEFAULT 0
*2 … přihlašovací jméno zaměstnance (varchar 5) nebo přihlašovací jméno osoby (varchar 7)
*3 … počet osob > 0 & počet osob < 100 a zároveň také SUM (pocet) GROUP BY id_termin <= 100

Tabulka	Atribut	Datový typ	Délka	Klíč	Null	Index	IO	Význam, poznámka
Omezeni	id_omezeni	int		A	N			identity(1,1)
	datum_od	datetime		N	N	A	*1	datum počátku omezení
	datum_do	datetime		N	N	A	*1	datum ukončení omezení
	popis	varchar	150	N	N			popis důvodu omezení

*1 … datum musí být následujícího formátu: dd.MM.yyyy

Tabulka	Atribut	Datový typ	Délka	Klíč	Null	Index	IO	Význam, poznámka
Organizace	id_organizace	int		A	N			identity(1,1)
	nazev	varchar	50	N	N	A		název organizace
	typ	varchar	30	N	A	A		typ organizace
	ulice	varchar	25	N	A			název ulice
	cp	varchar	10	N	A			číslo popisné
	mesto	varchar	25	N	A			město
	psc	varchar	6	N	A		*1	poštovní směrovací číslo

*1 … PSČ musí být platného tvaru, např.: XXX XX nebo XXXXX

Tabulka	Atribut	Datový typ	Délka	Klíč	Null	Index	IO	Význam, poznámka
Osoba	id_osoba	varchar	7	A	N			přihlašovací jméno
	id_role	tinyint		N	A			cizí klíč Uzivatelska_role
	titul_pred	varchar	12	N	A			titul před jménem
	jmeno	varchar	15	N	N	A		jméno
	prijmeni	varchar	25	N	N	A		příjmení
	titul_za	varchar	10	N	A			titul za jménem
	ulice	varchar	25	N	A			název ulice
	cp	varchar	10	N	A			číslo popisné
	mesto	varchar	25	N	A			město
	psc	varchar	6	N	A		*1	poštovní směrovací číslo
	telefon	char	13	N	A	A	*2	telefonní číslo
	email	varchar	35	N	A		*3	emailová adresa

*1 … PSČ musí být platného tvaru, např.: XXX XX nebo XXXXX
*2 … telefonní číslo musí odpovídat platnému předpisu pro telefonní číslo v ČR, pro validaci je použit následující regulární výraz: (\+420|420)? ?[1-9]{1}\d{2} ?\d{3} ?\d{3}

3 … emailová adresa musí odpovídat platnému předpisu, pro validaci byl použit následující regulární výraz: \w+([-+.']\w+)@\w+([-.]\w+)*\.\w+([-.]\w+)*

Tabulka	Atribut	Datový typ	Délka	Klíč	Null	Index	IO	Význam, poznámka
Osoba_Organizace	id_organizace	int		A	N			cizí klíč Organizace
	id_osoba	varchar	7	A	N			cizí klíč Osoba

Tabulka	Atribut	Datový typ	Délka	Klíč	Null	Index	IO	Význam, poznámka
Porad	id_porad	int		A	N			identity(1,1)
	id_misto	tinyint		N	N			cizí klíč Misto
	id_kategorie	int		N	N			cizí klíč Kategorie
	nazev	varchar	50	N	N	A		název pořadu
	delka	smallint		N	A		DEFAULT 0	délka pořadu v min.
	cena_deti	smallint		N	A			cena pro děti v Kč
	cena_dospeli	smallint		N	A			cena pro dospělé v Kč
	popis	varchar	500	N	N			anotace pořadu
	zkratka	char	4	N	N		*1	zkratka pořadu
	odstraneno	bit	1	N	N	A	*2	platnost záznamu?

*1 … systémová zkratka musí být v rozsahu 4 znaků složená z malých a velkých písmen nebo číslic … UNIQUE constraint … zkratka musí být unikátní

*2 … hodnota True/False … DEFAULT 0 … indikuje platnost záznamu v systému … pokud chce uživatel smazat záznam o pořadu ze systému, nastaví se tento atribut na hodnotu True a pořad se odebere z aktuálních nabídek jednotlivých funkcí systému

Tabulka	Atribut	Datový typ	Délka	Klíč	Null	Index	IO	Význam, poznámka
Poznamka	id_poznamka	int		A	N			identity(1,1)
	id_osoba	varchar	7	N	N			cizí klíč Osoba
	text	varchar	150	N	N			obsah poznámky

Tabulka	Atribut	Datový typ	Délka	Klíč	Null	Index	IO	Význam, poznámka
Program	id_program	int		A	N			identity(1,1)
	id_termin	int		N	N	A		cizí klíč Termin
	id_porad	int		N	N	A		cizí klíč Porad
	id_zamestnanec	varchar	5	N	A			cizí klíč Zamestnanec
	popis	varchar	100	N	A			popis

Tabulka	Atribut	Datový typ	Délka	Klíč	Null	Index	IO	Význam, poznámka
Rezervace	id_rezervace	int		A	N			identity(1,1)
	id_program	int		N	N	A		cizí klíč Program
	id_osoba	varchar	7	N	N	A		cizí klíč Osoba
	id_organizace	int		N	A	A		cizí klíč Organizace
	pocet_mist	tinyint		N	N		*1	počet míst
	datum_rezervace	datetime		N	N	A	*2	datum rezervace

*1 … počet míst > 0 & počet míst < 80 a zároveň také SUM (pocet_mist) GROUP BY id_termin <= 80
*2 … datum přidání rezervace musí být následujícího formátu: dd.MM.yyyy

Tabulka	Atribut	Datový typ	Délka	Klíč	Null	Index	IO	Význam, poznámka
Sluzba	id_sluzba	int		A	N			identity(1,1)
	id_zamestnanec	varchar	5	N	N	A		cizí klíč Zamestnanec
	datum	datetime		N	N	A	*1	datum služby

*1 … datum služby musí být následujícího formátu: dd.MM.yyyy

Tabulka	Atribut	Datový typ	Délka	Klíč	Null	Index	IO	Význam, poznámka
Stav	id_stav	tinyint		A	N			identity(1,1)
	nazev_stavu	varchar	20	N	N			název stavu obj.

Tabulka	Atribut	Datový typ	Délka	Klíč	Null	Index	IO	Význam, poznámka
Termin	id_termin	int		A	N			identity(1,1)
	zacatek	datetime		N	N	A	*1	počáteční datum
	konec	datetime		N	N		*1	koncové datum
	den	varchar	10	N	N	A	*3	název dne v týdnu
	uzaverka	bit	1	N	N	A	*2	byla provedena uzávěrka?
	tisk	bit	1	N	N		*2	byl proveden tisk?

*1 … počáteční i koncové datum musí být následujícího formátu: dd.MM.yyyy hh:mm:ss a zároveň datum začátku a konce termínu se nesmí shodovat s žádným záznamem v tabulce Omezení
*2 … hodnota True/False … DEFAULT 0
*3 … Pondělí, Úterý, Středa, Čtvrtek, Pátek, Sobota, Neděle

Tabulka	Atribut	Datový typ	Délka	Klíč	Null	Index	IO	Význam, poznámka
Uzivatelska_role	id_role	tinyint		A	N			identity(1,1)
	nazev_role	varchar	20	N	N			název uživ. role
	popis	varchar	50	N	A			detailní popis

Tabulka	Atribut	Datový typ	Délka	Klíč	Null	Index	IO	Význam, poznámka
Zamestnanec	id_zamestnanec	varchar	5	A	N			přihlašovací jméno
	id_role	tinyint		N	N			cizí klíč Uzivatelska_role
	titul_pred	varchar	12	N	A			titul před jménem
	jmeno	varchar	15	N	N	A		jméno
	prijmeni	varchar	25	N	N	A		příjmení
	titul_za	varchar	10	N	A			titul za jménem
	ulice	varchar	25	N	A			název ulice
	cp	varchar	10	N	A			číslo popisné
	mesto	varchar	25	N	A			město
	psc	varchar	6	N	A		*1	poštovní směrovací číslo
	telefon	char	13	N	N		*2	telefonní číslo
	email	varchar	35	N	N		*3	emailová adresa
	zkratka	char	4	N	N		*4	systémová zkratka
	odstraneno	bit	1	N	N	A	*5	platnost záznamu?

*1 … PSČ musí být platného tvaru, např.: XXX XX nebo XXXXX

*2 … telefonní číslo musí odpovídat platnému předpisu pro telefonní číslo v ČR, pro validaci je použit následující regulární výraz: (\+420|420)? ?[1-9]{1}\d{2} ?\d{3} ?\d{3}

3 … emailová adresa musí odpovídat platnému předpisu, pro validaci byl použit následující regulární výraz: \w+([-+.']\w+)@\w+([-.]\w+)*\.\w+([-.]\w+)*

*4 … systémová zkratka musí být v rozsahu 4 znaků složená z malých a velkých písmen nebo číslic … UNIQUE constraint … zkratka musí být unikátní

*5 … hodnota True/False … DEFAULT 0 … indikuje platnost záznamu v systému … pokud chce uživatel smazat záznam o zaměstnanci ze systému, nastaví se tento atribut na hodnotu True a zaměstnanec se odebere z aktuálních nabídek jednotlivých funkcí systému

C Datové toky

Datový tok	Atribut	Datový typ	Délka	Klíč	Null	Index	IO	Význam, poznámka
zakaznik	jmeno	varchar	15	N	N			viz. Osoba
	prijmeni	varchar	25	N	N			viz. Osoba
	telefon	char	13	N	N			viz. Osoba
nova_objednavka	zacatek	datetime		N	N			viz. Termin
	konec	datetime		N	N			viz. Termin
	jmeno	varchar	15	N	N			viz. Osoba
	prijmeni	varchar	25	N	N			viz. Osoba
	telefon	char	13	N	N			viz. Osoba
	nazev	varchar	50	N	A			viz. Organizace
	pocet	int		N	N			viz. Objednavka
	celkova_delka	int		N	A			viz. Objednavka
	nazev	varchar	30	N	N			viz. Kategorie
	prednaskovy_sal	bit	1	N	N			viz. Objednavka
	jednohubka	bit	1	N	N			viz. Objednavka
	nazev_filmu	varchar	50	N	A			viz. Objednavka
	nazev_jednohubky	varchar	50	N	A			viz. Objednavka
	mira	bit	1	N	N			viz. Objednavka
	hvezdarna	bit	1	N	N			viz. Objednavka
	poznamka	varchar	500	N	A			viz. Objednavka
	id_porad	int		N	A			viz. Objednavka
	id_zamestnanec	varchar	5	N	A			viz. Objednavka
detail_obj	zacatek	datetime		N	N			viz. Termin
	konec	datetime		N	N			viz. Termin
	jmeno	varchar	15	N	N			viz. Osoba
	prijmeni	varchar	25	N	N			viz. Osoba
	telefon	char	13	N	N			viz. Osoba
	nazev	varchar	50	N	A			viz. Organizace
	nazev	varchar	50	N	A			viz. Porad
	jmeno	varchar	15	N	A			viz. Zamestnanec
	prijmeni	varchar	25	N	A			viz. Zamestnanec

	pocet	int		N	N			viz. Objednavka
	celkova_delka	int		N	A			viz. Objednavka
	nazev	varchar	30	N	A			viz. Kategorie
	nazev_stavu	varchar	20	N	N			viz. Stav
	prednaskovy_sal	bit	1	N	N			viz. Objednavka
	jednohubka	bit	1	N	N			viz. Objednavka
	mira	bit	1	N	N			viz. Objednavka
	hvezdarna	bit	1	N	N			viz. Objednavka
	nazev_filmu	varchar	50	N	A			viz. Objednavka
	nazev_jednohubky	varchar	50	N	A			viz. Objednavka
	poznamka	varchar	500	N	A			viz. Objednavka
	autor	varchar	10	N	N			viz. Objednavka
obj	zacatek	datetime		N	N			viz. Termin
	konec	datetime		N	N			viz. Termin
	jmeno	varchar	15	N	A			viz. Zamestnanec
	prijmeni	varchar	25	N	A			viz. Zamestnanec
	zkratka	char	4	N	N			viz. Zamestnanec
	nazev	varchar	50	N	A			viz. Porad
	nazev	varchar	30	N	N			viz. Kategorie
	nazev	varchar	50	N	A			viz. Organizace
	pocet	int		N	N			viz. Objednavka
param_obj	zacatek	datetime		N	N			viz. Termin
	konec	datetime		N	N			viz. Termin
	prijmeni	varchar	25	N	N			viz. Osoba
	nazev	varchar	50	N	A			viz. Organizace
	nazev_stavu	varchar	20	N	N			viz. Stav
porad_info	nazev	varchar	50	N	N			viz. Porad
	nazev	varchar	30	N	N			viz. Kategorie
	nazev	varchar	30	N	N			viz. Misto
program	zacatek	datetime		N	N			viz. Termin
	konec	datetime		N	N			viz. Termin
	jmeno	varchar	15	N	A			viz. Zamestnanec

	prijmeni	varchar	25	N	A			viz. Zamestnanec
	zkratka	char	4	N	A			viz. Zamestnanec
	nazev	varchar	50	N	A			viz. Porad
	pocet_mist	tinyint		N	A			viz. Rezervace
rezervace	jmeno	varchar	15	N	N			viz. Osoba
	prijmeni	varchar	25	N	N			viz. Osoba
	telefon	char	13	N	N			viz. Osoba
	nazev	varchar	50	N	A			viz. Organizace
	pocet_mist	tinyint		N	N			viz. Rezervace
	datum_rezervace	datetime		N	N			viz. Rezervace
param_rez	zacatek	datetime		N	N			viz. Termin
	konec	datetime		N	N			viz. Termin
	prijmeni	varchar	25	N	N			viz. Osoba
	nazev	varchar	50	N	A			viz. Organizace
osobni_udaje	titul_pred	varchar	12	N	A			viz. Osoba, Zamestnanec
	jmeno	varchar	15	N	N			viz. Osoba, Zamestnanec
	prijmeni	varchar	25	N	N			viz. Osoba, Zamestnanec
	titul_za	varchar	10	N	A			viz. Osoba, Zamestnanec
	ulice	varchar	25	N	A			viz. Osoba, Zamestnanec
	cp	varchar	10	N	A			viz. Osoba, Zamestnanec
	mesto	varchar	25	N	A			viz. Osoba, Zamestnanec
	psc	varchar	6	N	A			viz. Osoba, Zamestnanec
	telefon	char	13	N	N			viz. Osoba, Zamestnanec
	email	varchar	35	N	N			viz. Osoba, Zamestnanec
novy_zakaznik	titul_pred	varchar	12	N	A			viz. Osoba
	jmeno	varchar	15	N	N			viz. Osoba
	prijmeni	varchar	25	N	N			viz. Osoba
	titul_za	varchar	10	N	A			viz. Osoba
	ulice	varchar	25	N	A			viz. Osoba
	cp	varchar	10	N	A			viz. Osoba
	mesto	varchar	25	N	A			viz. Osoba
	psc	varchar	6	N	A			viz. Osoba

	telefon	char	13	N	A			viz. Osoba
	email	varchar	35	N	A			viz. Osoba
	nazev	varchar	50	N	N			viz. Organizace
	typ	varchar	30	N	A			viz. Organizace
	ulice	varchar	25	N	A			viz. Organizace
	cp	varchar	10	N	A			viz. Organizace
	mesto	varchar	25	N	A			viz. Organizace
	psc	varchar	6	N	A			viz. Organizace